2008-12

# Improving Maritime Prepositioning Force (MPF) offloads using modeling and simulation

Thomas, Brandon K.

Monterey, California. Naval Postgraduate School


# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**IMPROVING MARITIME PREPOSITIONING FORCE (MPF) OFFLOADS USING MODELING AND SIMULATION**

by

Brandon K. Thomas

December 2008

Thesis Advisor: Don Brutzman
Second Reader: Terry Norbraten

**This thesis was done at the MOVES Institute**
**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** | **2. REPORT DATE** December 2008 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE**: Improving Maritime Prepositioning Force Arrival and Assembly Operations Using an Open Source Visualization Extensible Three-Dimensional Tool: Improving the MPF Offload Using X3D and Viskit | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR : Brandon K. Thomas** | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | | | **12b. DISTRIBUTION CODE** |


**13. ABSTRACT**

The Marine Corps' Maritime Prepositioning Force (MPF) marries fly-in troops to their gear in an expeditionary environment. The arrival and assembly operation underneath this larger umbrella of MPF Operations proves itself a somewhat chaotic, definitively complex and dynamic logistics operation. From the moment the offload of the ship or ships begins when equipment and rolling stock exit the ships, until it ends as using units sign for their intended equipment, all personnel involved in this process—drivers, assistant drivers, heavy equipment handlers, crane operators, equipment managers—and all equipment involved present a flurry of activity that must be effectively managed, tracked, and optimized. Modeling, Virtual Environments, and Simulation, or MOVES, tools, aid in providing such capability. The creation of a Discrete Event Simulation (DES) using the open-source tool Viskit enables MPF planning, training, and analysis in its ability to portray the effects that size, amount of personnel support, and time have on the operation. Scenario Authoring and Visualization for Advanced Graphical Environments (Savage) comprises an archive of extensible three-dimensional (3D) models that, when tied to the DES in an Extensible 3D (X3D) Graphics environment, enable the animation of the simulation, and when connected to real-world tracking data of the offload, allow for real-time visual tracking of this logistics process, creating a Common Operating Picture (COP) for the Arrival and Assembly Operations Group (AAOG).


| **14. SUBJECT TERMS** Maritime Prepositioning Force, Viskit, X3D Graphics, Operational Logistics, Modeling and Simulation | | | **15. NUMBER OF PAGES** 115 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**This thesis was done at the MOVES Institute**
**Approved for public release; distribution is unlimited**

# IMPROVING MARITIME PREPOSITIONING FORCE (MPF) OFFLOADS USING MODELING AND SIMULATION

Brandon K. Thomas
Captain, United States Marine Corps
B.S., United States Naval Academy, 1999

Submitted in fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL**
**December 2008**

Author: Brandon K. Thomas

Approved by: Don Brutzman
Thesis Advisor

Terry Norbraten
Second Reader

Mathias Kölsch
Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

The Marine Corps' Maritime Prepositioning Force (MPF) marries fly-in troops to their gear in an expeditionary environment. The arrival and assembly operation underneath this larger umbrella of MPF Operations proves itself a somewhat chaotic, definitively complex and dynamic logistics operation. From the moment the offload of the ship or ships begins when equipment and rolling stock exit the ships, until it ends as using units sign for their intended equipment, all personnel involved in this process—drivers, assistant drivers, heavy equipment handlers, crane operators, equipment managers—and all equipment involved present a flurry of activity that must be effectively managed, tracked, and optimized. Modeling, Virtual Environments, and Simulation, or MOVES, tools, aid in providing such capability. The creation of a Discrete Event Simulation (DES) using the open-source tool Viskit enables MPF planning, training, and analysis in its ability to portray the effects that size, amount of personnel support, and time have on the operation. Scenario Authoring and Visualization for Advanced Graphical Environments (Savage) comprises an archive of extensible three-dimensional (3D) models that, when tied to the DES in an Extensible 3D (X3D) Graphics environment, enable the animation of the simulation, and when connected to real-world tracking data of the offload, allow for real-time visual tracking of this logistics process, creating a Common Operating Picture (COP) for the Arrival and Assembly Operations Group (AAOG).

THIS PAGE INTENTIONALLY LEFT BLANK

# EXECUTIVE SUMMARY

Logistics processes are data-driven, time intensive, and chaotic by their very nature. A Logistics Operations Center (LOC) hosts a flurry of activity—from meeting and filling support requests to determining requirements to anticipating unexpected shortfalls, all the while maintaining a scalable economy of force and supplies—in which redundancies must be avoided. The Arrival and Assembly of Maritime Prepositioning Force (MPF) equipment and supplies represents such a logistics process.

The Maritime Prepositioning Force (MPF) is the force in readiness for Navy and Marine Corps operational logistics. It consists of approximately sixteen ships stationed at three equidistant locations throughout the globe, packed with all of the equipment necessary to support either Marine Expeditionary Units (MEU), Marine Expeditionary Brigades (MEB), or even an entire Marine Expeditionary Force (MEF) for a period of up to sixty days, long enough for follow-on shipping to catch up to the support offered by initially prepositioned equipment. In support of any exercise or operation from anywhere along the spectrum of conflict, from humanitarian assistance to total war, the MEF marries fly-in troops to the aforementioned prepositioned equipment in a matter of days, as opposed to the weeks or possibly months it takes otherwise. In addition to this, the in- or close-to-theatre marriage of troops with equipment has the potential for significantly lightening the load for existing deployments in which gear accompanies personnel for the entire deployment.

The Arrival and Assembly of MPF equipment consists of the offload of the ship or ships, and after the equipment is no longer needed, its regeneration. In the interest of scope, the MPF offload serves as the primary focus of effort in this endeavor to improve the system as a whole. Currently, software support for tracking MPF equipment and personnel inside the Arrival and Assembly Operations Group (AAOG), at best consists of Microsoft Excel pivot tables, which only output percentages of what kind of equipment is where in this logistics process. Questions are answered in the form of data and tables and charts, and as a result, the requestor requires an analytic learning curve, rather than the intuitive visual insight a visual display of such needed information could provide.

Viskit, a Java-based, user-friendly, open source program implementing a Discrete Event Simulation (DES), provides a way in which a user can essentially "draw" event graphs that simulate the movement of things and/or people through a given process. In this case, the DES of the MPF Offload is built, following each Principle End Item (PEI) from the second it begins its offload from the ship itself to the second that it is signed for by its using unit. The number of PEI's to be offloaded from a particular ship, the numbers of personnel available at each logistics process and available to transition the PEI's from logistics process to logistics process, and the time and associated variability it takes to accomplish each task are all parameters that can be manipulated to change the dimensions of the process as a whole. The overall time it takes to offload the ship can offer insights into analytic approximations of how many people may be determined to be used to accomplish the offload. Not only does this simulation prove invaluable in Arrival and Assembly planning, but it can also be used in training, in the MPF Staff Planner's course, for one, and in conducting an analysis of the execution, matching up the tracking of the actual exercise to a simulation of the same offload, as a means to offer possible insight into potential shortfalls in the process itself.

The X3D-Edit authoring tool serves as a means to generate a 3D environment in which the aforementioned simulation can be visually animated. Animation classes called "mover classes" written in Java enable the DES to come to life using the IEEE Distributed Interactive Simulation (DIS) protocol. This virtual environment in which X-Y-Z coordinates are standardized allows this local simulation to become web-based. Such potentially global visibility of this local process enables organizations worldwide to view such an integrated operation. In its simplest application, however, this 3D visualization of the offload as a logistics process is the Common Operating Picture (COP) that has since then been missing from the AAOG. The ability to visually track equipment, supplies, and personnel as they traverse through this logistics process is essential to enabling situational awareness in the operations center.

A future that is more joint, more collaborative, and more multi-national and multi-lingual can be greatly aided by the inherent and universal communication that visualization provides. Especially in an environment grounded in logistics, where time is

of the essence and where situational awareness helps greatly in maximizing the throughput of equipment needed to accomplish the mission, the visualization of logistics processes is crucial. Such a direction in technology can only serve to make the tedious, laborious, and mind numbing work of logistics more predictable and effective.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF FIGURES

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| 3D | 3 Dimensional |
| AAAV | Advanced Amphibious Assault Vehicle |
| AAOE | Arrival and Assembly Operations Element |
| AAOG | Arrival and Assembly Operations Group |
| ADV | Advanced |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| BIC | Blount Island Command |
| BOG | Beach Operations Group |
| C2PC | Command and Control Personal Computer |
| COP | Common Operating Picture |
| COC | Combat Operations Center |
| CSS | Combat Service Support |
| CSSOC | Combat Service Support Operations Center |
| DES | Discrete Event Simulation |
| Diskit | Distributed Interactive Simulation Kit |
| DOE | Design of Experiments |
| DOM | Document Object Model |
| DPR | Daily Process Report |
| EAF | Expeditionary Airfield |
| EDS | Electronic Data Systems |
| FAQ | Frequently Answered Questions |

| | |
|---|---|
| EWTGLANT | Expeditionary Warfare Training Group, Atlantic |
| EWTGPAC | Expeditionary Warfare Training Group, Pacific |
| GCCSM | Global Command and Control System |
| GUI | Graphical User Interface |
| HMMWV | High-Mobility Multi-purpose Wheeled Vehicle |
| HTML | Hypertext Markup Language |
| ICODES | Integrated Computer Development System |
| JAXB | Java Architecture for XML Binding |
| JLTI | Joint Limited Technical Inspection |
| LARC | Lighter Amphibious Re-supply Cargo |
| LFSP | Landing Force Support Party |
| LMSR | Large, Medium Speed, Roll on/Roll off |
| LOC | Logistics Operations Center |
| LTM | Last Tactical Mile |
| LVS | Logistics Vehicle System |
| M&S | Modeling and Simulation |
| MAGTF | Marine Air Ground Task Force |
| MCC | Movement Control Center |
| MCPIC | Marine Corps Prepositioning Information Center |
| MDSSII | MAGTF Deployment Support System II (See MAGTF) |
| MPF | Maritime Prepositioning Force |
| MPS | Maritime Prepositioning Ship |
| MPSRON | Maritime Prepositioning Squadron |
| MTVR | Medium Tactical Vehicle Replacement |

| | |
|---|---|
| NALMEB | Norway Air-landed Marine Expeditionary Brigade |
| NAVMC | Navy Marine Corps |
| NCO | Non-commissioned Officer |
| NPS | Naval Postgraduate School |
| OPP | Offload Preparation Party |
| OSS | Open Source Software |
| PES-V | Prepositioned Equipment and Supply Viewer |
| PCL | Property Change Listener |
| PEI | Principal End Item |
| POG | Port Operations Group |
| PPS | Prepositioned Planning System |
| PDF | Probability Distribution Function |
| RO | Responsible Officer |
| RTCH | Rough Terrain Container Handler |
| SA | Situational Awareness |
| SAVAGE | Scenario Authoring and Visualization for Advanced Graphical Environments |
| Simkit | Simulation Kit |
| SMAL | Savage Modeling and Analysis Language (See SAVAGE) |
| SNCO | Staff Non-commissioned Officer |
| TAAT | Technical Acquisition Assistance Team |
| TAMCN | Table of Allowance Materiel Control Number |
| T/E | Table of Equipment |
| T/O | Table of Organization |

| | |
|---|---|
| USNS | US Naval Ship |
| UURI | Using Unit Responsible Items |
| Viskit | Visual Kit |
| VRML | Virtual Reality Modeling Language |
| W3C | Web3D Consortium |
| X3D | Extensible 3D Graphics |
| XML | Extensible Markup Language |
| XSLT | Extensible Stylesheet Language for Transformations |

# ACKNOWLEDGMENTS

I would like to extend sincere gratitude to the following people who played an integral role in the successful completion of this thesis:

Thank you to my Thesis Advisor, Dr. Don Brutzman, for guidance and support throughout the conduct of this thesis research and work.

Thank you to my Co-Advisor, Col Ed Lesnowicz (Ret), and my Second Reader, Terry Norbraten, for their tremendous help and guidance in the successful completion and accomplishment of this thesis. I extend an especial thanks to Professor Arnie Buss for his time and assistance in the creation of the DES in Viskit.

Thank you to Installations and Logistics, U.S. Marine Corps, for supporting travel to the Royal Naval Base in Aqaba, Jordan, to observe Exercise Native Fury, conducted by the 13th MEU, where I was able to validate the current MPF Offload process and was meet with representatives from BIC, as well as FSR's from current Simulations Programs used by the Marine Corps. I extend an especial thanks to LCDR McKenna for his profound wisdom relating to the current state of MPF Operations from a strategic and overarching perspective.

Thank you to my professors at the Naval Postgraduate School who brought me to where I am in regards to technical proficiency in Modeling and Simulation, Optimization, Web-based Simulation, and Cognitive Agents and Modeling.

Thank you to my friends for their love and support, the staff of the Yoga Sanctuary, the Indoor Forest Theatre, the Carmel Mission Basilica, and many, many others.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION AND PROBLEM STATEMENT

## A. OVERVIEW

The Maritime Prepositioned Force (MPF), as its name implies, constitutes a fleet of ships prepositioned throughout the world in support of tactical, operational, and strategic needs of the Marine Corps and US and allied forces. According to the Marine Corps Warfighting Publication 3-32 entitled Maritime Prepositioning Force Operations, the MPF Marine Air Ground Task Force (MAGTF) "directly support[s the] national maritime strategy of protecting key naval chokepoints and sea lines of communications," as well as responding to crises and contingencies throughout the globe.

Currently, sixteen civilian ships house hundreds of containers, pieces of rolling stock, and thousands of equipment items that, when called for, are intelligently married, primarily to Marines and secondarily to other services' forces near the area where they are needed. This concept allows for a faster buildup of a larger footprint of military support resources in needed areas of the world, and serves as a step closer in operational logistics to an always-ready military presence around the world.

The role of the Maritime Prepostitioned Force is becoming a more viable option for contingencies today. The Marine Expeditionary Unit, or MEU, accompanies the world's Expeditionary Strike Groups, or ESG's (more commonly referred to previously as the Amphibious Ready Group, or ARG), the smallest arm of the Marine Air Ground Task Force, or MAGTF, and the President's first choice in global crisis response. It may begin to prove more costly, having to carry all of the gear, rolling stock, and equipment needed with them on the ships with which they are traveling, than simply falling in on gear from the MPF. Figure 1 illustrates these relationships.

Though originally designed to support a MEB, or Marine Expeditionary Brigade, the MPF was used in the largest Arrival and Assembly operation in history with eleven ships of the entire MPF fleet used in the buildup of forces in Operation Iraqi Freedom. Exercise Native Fury tested the capability of the 13$^{th}$ MEU with the partial offload of the USNS Obregon, an MPF Arrival and Assembly exercise. As a result of this increasing viability, feasibility, and diversification in how the MPF may need to be off-loaded and

regenerated, the need for automated systems that track, visualize, and enable more thorough and better analysis become increasingly important as well.

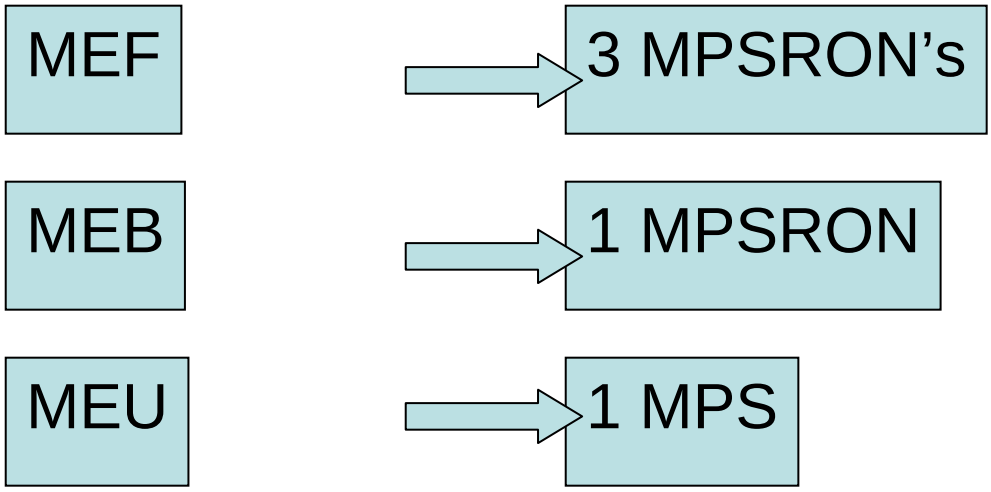



Table 1. This simple chart delineates the requirements to support each particular sized Marine Corps entity for a given conflict.

The potential future for the Marine Prepositioned Force embodies a fleet of ships able to pull into permissive ports—ports whose governments permit MPF usage for a given operation or exercise—across the globe and be offloaded for not one, but multiple smaller contingencies simultaneously, with different weight being given to different missions and efforts, and those ships being offloaded. Tailored global tactical response aided and enabled by the MPF can allow the least intrusive means to accomplish a mission on the ground, and potentially enable the least costly execution as well.

## B. MOTIVATION

The Marine Corps defines Logistics as "the science of maximizing throughput."

Logistics can be thought of as simply getting the right things from point A to point B. A large quantity of entities, anything from parts to food to vehicles, is likened to grains of sand in an hourglass. The middle of the hourglass is the choke point, through which all of those entities must pass. The job of the logistician is to manage goods and services in such a way as to maximize the passing of those metaphorical "particles of sand" through each choke point, otherwise known as maximizing throughput.

Modeling and Simulation, M&S, provides a tremendous tool capability that, when applied correctly, can more quickly process those grains of sand. In this case, those grains are information, the data retrieved in Logistics Operations Centers (LOC), in Beach Operations Groups (BOG), and in Port Operations Groups (POG) around the globe within the Marine Corps.

Were one to walk into a Logistics Operations Center today, the first thing that catches his/her attention would be status boards: boards for current operations, boards for future operations, which convoys are out, where are they, who is on them, what is slated to go out, vehicle status boards, personnel status boards, accountability, status, and plans and schedules. A trained eye can walk into such an operations center and for the most part determine what is going on, yet still this requires mounds of expertise coupled with time, no matter how experienced the individual may be. The reality of logistics is that it can be tedious, sometimes mind-numbing work that can always be done more smartly, more efficiently, and can be greatly aided by the use of programs that make some of these detailed processes automatic. In logistics, computer support is essential for managing large operations.

To capture this information into a useable format (the ever broad and transmutable XML) and then use that converted information to create simulations, models, and/or visual aids, does not merely help to transform miles of cryptic data into visually intuitive forms. Progress and problems become measurable, predictable, and evident so that alternative plans and corrective actions can be pursued. Comparing repeatable simulations with real world results also has the potential to change how logisticians view logistics: not only as numbers to be crunched, but also as the ever-moving, ever-evolving field that it is.

A problem facing many working logisticians is that too much detailed work needs to be done, precluding opportunities to step back and figure out how to work more smartly. Also in this field, advanced technical expertise is not always a primary user skill, so any support system provided to help must be affordable and user friendly. The solutions explained in this work are intended to be inexpensive, repeatable, scalable, and useable by operating forces.

## C. SCOPE

The main thrust of the thesis is oriented towards the automation of a Port Operations Group in support of an amphibious raid. Future work might improve existing simulations at Expeditionary Warfare Training Group, Atlantic, or EWTGLANT, for such training, if there are any. The simulation includes a set of 3D models that contain the operational transition from ship to shore, to include the use of LCAC's (Landing Craft Air Cushion), LCU's (Landing Craft Unit), lighterage (floating docks), etc.

The figure below depicts a row of vehicles in the Arrival and Assembly process for such an MPF Offload. Though the scope of this thesis is limited to this one particular form of logistics operation, the same methodology and process can be applied to any number of logistics processes, to include but not limited to convoy operations and Beach Operations Groups, or BOG's.

Figure 1. USMC vehicles wait to be driven from the port to the Movement Control Center, Agami, Egypt, Operation Bright Star, Nov '01.

## D. RESEARCH QUESTIONS

Can Modeling and Simulation be used to improve Logistics Systems in the Marine Corps?

-Can the "XML-ization" of great quantities of integral logistics data be harnessed and used to perform a variety of tasks that might aid in the better visualization, training, and perhaps eventual analysis to improve the manner in

which logisticians in the Marine Corps currently interact with that information, and perhaps might be able to help logisticians make better decisions?

-Can the combination of Web-based Simulation with Artificial Intelligence be used to create logistics convoys based on all available information, to include support requests?

-Can XML be used to aid in bringing more closely together RFID technologies and the Supply and Maintenance System (SASSY)?

-Can Modeling and Simulation, in particular agent-based simulation, be used in conjunction with Optimization techniques to improve the maximization of throughput?

-Can Modeling and Simulation be used, not merely as a means to improve the systems themselves, but also as a real-time visual aid to improve the overall situational awareness of Operations Centers and respective personnel?

-Can enough systems in Operations Centers be automated to better economize the strength of the Marine Corps war-fighting force, so that more Marines can be used at the tip of the spear rather than managing the details of the data analysis?

-Can modeling and simulation be used in such a manner to perform simple convoy planning as it relates to JIEDDO (Joint Improvised Explosive Device Defeat Office)?

Thus many important questions are considered. Proof of concept capabilities are demonstrated wherever possible in order to establish the basis for creating systematic deployable solutions.

## E. ORGANIZATION OF THESIS

### 1. MPF Operations, Background, and Related Work

The second chapter of this thesis describes the constitution of the Maritime Prepositioning Force, as well as information concerning the equipment it carries, and existing systems used to manage the information concerning the makeup of the ships.

Arrival and Assembly operations are discussed in detail, and the means by which operators and commanders attempt to maintain open lines of communication and effective information flow through their operations centers is also considered. Existing programs and systems that are currently on the market today that attempt to improve logistics systems are considered at the end of this section.

### 2. Simulation Tools

Simulation tools used in this thesis are discussed in the third chapter of this thesis. The foundational language of Java upon which the simulations are built are delineated from the ground up, beginning with a discussion of the methodology used to achieve the desired results. The languages used in this simulation are discussed, with extensive information regarding the making of a discrete event simulation. At the end of this section, other means to improve this particular logistics system are also discussed, namely the fields of optimization and artificial intelligence.

### 3. Applying the Simulation Tools to the Logistics Operation

The fourth chapter of this thesis applies the simulation tools discussed in the third chapter to the logistics operation discussed in the second chapter. The process of using modeling and simulation to improve logistics systems in general is considered, including (1) building a Viskit Discrete Event Simulation of the given process, (2) building a complex adaptive system design of the given process, (3) constructing an optimization formulation or network graph of the given process, (4) converting data into XML, (5) capturing relevant terrain data in X3D, to include ports, roads, and camps, (6) building or using those relevant models in an X3D environment, (7) putting it all together in a visualization, and (8) capturing relevant statistics and comparing the results. Extreme programming is introduced, as well as the base scenario and second more complex scenario are illustrated and discussed.

### 4. Simulation Results and Analysis

The next chapter deals with the analysis of the results achieved by the simulation tools used in the previous chapter. Time Distribution considerations are discussed, as well as other design of experiments (DOE) considerations.

## 5. Conclusions, Recommendations for Future Work, and Appendices

The rest of the thesis relays the conclusions reached through the process of creating a model and simulation of this particular logistics process. The appendices discuss open source considerations, show the Analyst Report generated by Viskit, and consider a security-networking problem using the tenets of optimization in the same type of operation.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. MPF OPERATIONS BACKGROUND AND RELATED WORK

## A. INTRODUCTION

In order to understand how modeling and simulation can improve the MPF Arrival and Assembly Process, the concept behind the MPF must be first understood, as well as the logistical process upon which whose improvement is based.

## B. MAKEUP OF THE MPF FORCE

### 1. MPF Squadrons

The MPF contains three squadrons called MPSRON's, and a total of approximately sixteen ships, comprising three different classes of ships. They either have a RO/RO (Roll on Roll off) capability or a combination of that and a LO/LO (Lift on Lift off) capability. Normally, simultaneous offload between LO/LO and RO/RO helps to maximize throughput. LO/LO is achieved with the use of one or more cranes, shipboard or deckside. Each of the three MPF Squadrons supports the Marine Corps and is stationed in the Mediterranean Sea, at Diego Garcia, and in the Pacific around Guam, respectively. Currently, Blount Island Command (BIC) at Orlando, Florida, to equip and maintain the ships and cargo, contracts Honeywell.

The following list provides essential logistics information of MPF ships in the fleet. Pictures depict examples of each type of MPF ship in the current fleet.

| Post Operation Iraqi Freedom and MMC-8 | | | |
|---|---|---|---|
| CLASS | SHIP | NAME | Flag |
| **MPS-1 / MPSRON ONE : MEDITERRANEAN** | | | |
| AMSEA | AK 3008 | 2ND LT JOHN P. BOBO | * |
| AMSEA | AK 3009 | PFC D. T. WILLIAMS | |
| Maersk | AK 3001 | PFC WILLIAM B. BAUGH | |
| Waterman | AK 3006 | PFC E. A. OBREGON | * |
| MPF(E) | AK 3016 | LCpl Roy Wheat | |
| **MPS-2 / MPSRON TWO : Diego Garcia** | | | |
| AMSEA | AK 3012 | SGT W. R. BUTTON | * |
| AMSEA | AK 3010 | 1ST LT B. LOPEZ | |
| Maersk | AK 3004 | PVT FRANKLIN J. PHILLIPS | * |
| Waterman | AK 3005 | SGT MATEJ KOCAK | |
| MPF(E) | AK 3017 | Gysgt Fred W. Stockham | |
| **MPS-3 / MPSRON THREE : Guam** | | | |
| AMSEA | AK 3011 | 1ST LT JACK LUMMUS | * |
| Maersk | AK 3002 | PFC J. ANDERSON, JR | |
| Maersk | AK 3003 | 1ST LT A. BONNYMAN | |
| Maersk | AK 3000 | CPL L. HAUGE, JR | * |
| Waterman | AK 3007 | MAJ S. W. PLESS | |
| MPF(E) | AK 3015 | 1st Lt Harry L. Martin | |

Table 2. This list of MPF ships specifies in which squadron they currently reside.

### 2. MPF Tailoring Program

The current MPF is going through a Tailoring Program, a transition, in which Large, Medium Speed, Roll on/Roll off vessels, or LMSR's, from the Army are being procured, modified, and added to the fleet. They are incrementally replacing the Maersk Class Ships. The following is an example of such an LMSR:

Figure 2. This is a picture of the USNS Obregon taken at the Royal Jordanian Naval Base in Al'Aqaba, Jordan, July 2008.

### 3. Equipment

This includes everything from communications equipment to engineering equipment to transportation equipment to containers filled with virtually anything. It also includes exceptional equipment, such as lighterage (floating barges), Expeditionary Airfields (EAF), Fleet Hospitals, Bridging Equipment, etc.

Figure 3. On the left, a Gantry Crane Lifts a 20' CONNEX Container. On the right, HMMWV's are loaded in the belly of the USNS Obregon.

Figure 4. On the left, in the belly of the USNS Obregon are parked rolling stock, to include AAAV's. On the right, a LARC rests in the belly of the USNS Obregon.

### 4. Information Systems

#### *a. MAGTF Deployment Support System II Data*

MDSSII data contains the information regarding the equipment on the MPF ships. This information is stored by deck, each deck of the respective ship, and includes the information on associated mobile loads of the ship. The Marine Corps maintains a list of what is on the ship using a program written as an Excel spreadsheet, called MDSSII Data. They also use a ship-loading program called ICODES that shows where each piece of gear is loaded by deck, and frame. The using units determine how many and what pieces of gear are to be offloaded and given to the using units based on mission requirements. This could contain the entire ship, or just certain principal end items that need to be offloaded for a given exercise or operation. This could require, moreover, more than one MPF ship.

#### *b. Marine Corps Prepositioning Information Center*

The Marine Corps Prepositioning Information Center, MCPIC, is a web-based initiative to unite currently disparate information sources at one site for the prepositioning community; this includes but is not limited to, ship and squadron plans, prepositioning objectives, Tables of Equipment (T/E's), data of equipment and supplies

actually loaded on the MPS vessels, and other reference information. The PES-V allows the user to query actual ships' data through a variety of user-friendly methods and avenues.  Two of the major features of MCPIC are the Prepositioned Planning System (PPS), and the Prepositioned Equipments and Supplies Viewer (PES-V). The PPS features all squadrons' and ships' respective plans, associated reference data, and prepositioning objectives. It also provides information on Using Unit Responsibility Item (UURI) TAMCN's, as well as parent and child associations. The accessed data can be exported to either an Excel spreadsheet or text file.  The PES-V and its associated Deployment Workbench provides all available data for deployed MPS ships, that is, it provides a reflection of that which is actually loaded on the ships. PES-V enables the user to query available data in a variety of methods and it can quickly provide most users with the data they need.

Figure 5. This GUI portrays PES-V, a feature of MCPIC that enables the user to query actual ships' data through a variety of user-friendly methods and avenues.

Additionally, there is an Ad Hoc Reporting feature that allows for more advanced data queries. In addition to the above, MCPIC features a References & Information menu option. The References & Information menu option provides the following additional menu options: a prepositioning forum, allowing users to ask questions, submit topics of discussion, and provide subsequent comments; collateral information, currently providing Daily Process Reports (DPR's) for each ship; Frequently Answered Questions (FAQ's); and publications relating to prepositioning.

### *c. Integrated Computer Development System*

ICODES is a decision-support system that applies the Integrated Cooperative Decision-Making (ICDM) framework to the area of ship stow planning. It is designed to satisfy the focused stow planning demand of the U.S. Marine Corps and the U.S. Army by assisting personnel at the port to react quickly and efficiently to changing transportation requirements. As a shipload planning software tool, ICODES utilizes artificial intelligence (AI) principles and techniques to assist embarkation specialists in the rapid development of cargo stow plans.

Ship stow planning is the process of choreographing the banner by which equipment and supplies are loaded and unloaded from cargo ships. It shares many characteristics common to complex problem situations including the following characteristics: information overload, multiple interrelationships, uncertainty, layers of time constraints, and multiple decision makers needing a common means to collaboratively reach consensus. Figure 6 shows an ICODES entity of a typical agent warning status.

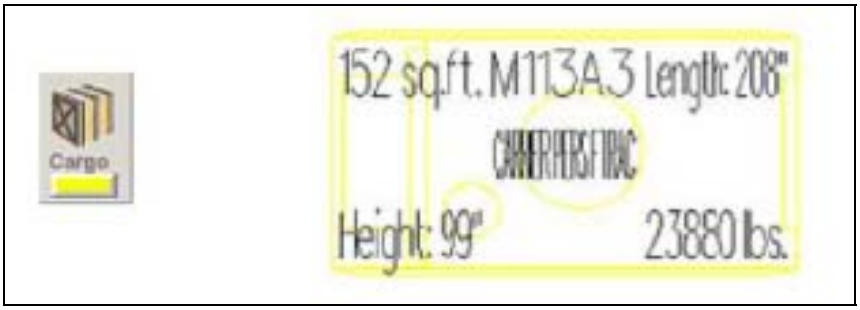



Figure 6. This is an ICODES entity GUI screenshot of a typical agent warning status.

ICODES incorporates expert computer agents that offer solutions during various phases of the planning process. These agents have knowledge in specific domains (e.g., cargo placement, hazardous materials handling, trim and stability impact, and accessibility) and evaluate and propose loading alternatives and recommendations. They reason and interact with each other as well as with the most important agent: the human

decision maker. The agents create a partnership and collaborate with expert human staff members during the various stages of the stow planning process.

In the last several years, ICODES has supported the Department of Defense by effectively controlling and maintaining the movement of cargo throughout the world. The Collaborative Agent Design Research Center (CADRC) provides system development, consulting, and training to military transportation planners and also maintains a 24-hour support team. The ICODES Version 5.0, released in February 2001 and installed worldwide in ports used by the U.S. Army, is able to develop stow plans for up to four ships concurrently. Utilizing the advanced technology embodied in the Integrated Cooperative Decision Making (ICDM) development framework, ICODES reduces a process that once took two experienced stow planners anywhere from two days to several hours. The U.S. Department of Defense has selected ICODES as the system of record (i.e., migration system) for ship stow planning. ICODES is currently being fielded to serve the specific embarkation needs of the U.S. Marine Corps within the concept of a Joint Deployment Community.

## C. ARRIVAL AND ASSEMBLY OPERATIONS

Underneath the umbrella of MPF Operations, the Arrival and Assembly process of offloading all necessary gear and equipment for the purposes of supporting an operation or exercise proves the center of gravity inside this complex logistical operation. In most cases, the MPF offload occurs at a permissive port, that is, a port that a given host country gives permission to use. In the case below, the Jordanian government allowed a Marine Corps contingent to perform an MPF exercise at their base in Al'Aqaba.

Figure 7. This is an aerial view of the Royal Jordanian Naval Base in Al'Aqaba. Joran, taken from a plane in July 2008.

As each Principle End Item, or PEI, rolls off of the ship, it enters the Port Operations Group (POG) where a Joint Limited Technical Inspection (JLTI) is conducted. If the PEI fails the JLTI, it stays at the POG and is put aside for maintenance. If the maintenance issue is severe enough, another PEI is selected from the ship to replace it. If not, it is serviced and proceeds out of the POG.

The idea behind MPF is that when military force is needed for an exercise or for an operation, Marines can be flown into an area, the MPF ships needed for the event pull into a nearby permissive port, and personnel and gear are married up in country or nearby.

Once the PEI leaves the POG, it enters a Disassociation Lot. Here the proper PEI's are associated with the correct equipment. That is to say, for example, a kit that is loaded onto a HMMWV may not be intended for the same using unit, the unit that will be using that particular piece of gear for the particular operation or exercise. All remaining equipment remains at the Disassociation Lot, waiting for the equipment and PEI's to come back so that it can be loaded back onto the ship in the same way.

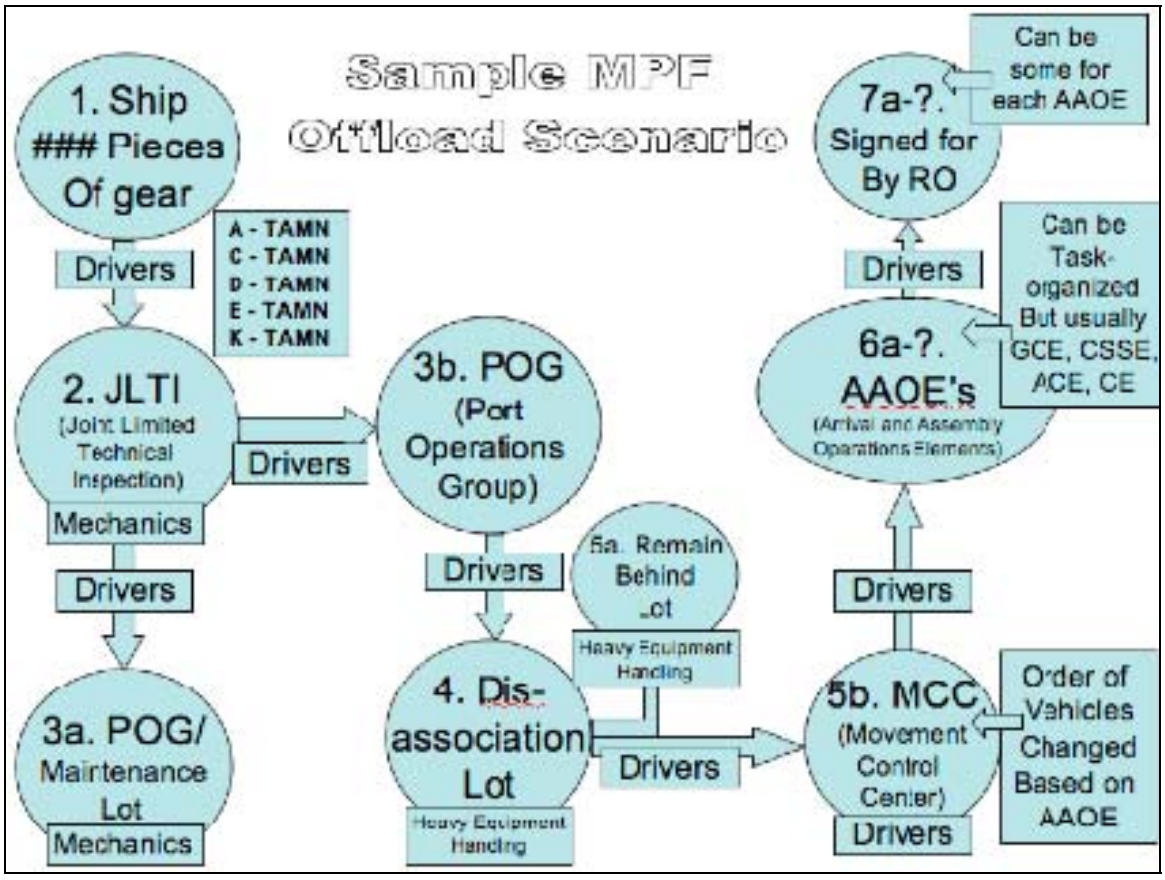



Figure 8. This flow chart tracks rolling stock movement throughout the process of an MPF offload.

After PEI's have been processed through the Disassociation Lot, they are sent to the Movement Control Center, or MCC. At the MCC, they are broken down into convoys based on the using unit to which they are intended. Then the convoys are dispatched to their particular AAOE, or Arrival and Assembly Operations Element, which might be further broken down into smaller using units.

Once the convoy reaches its destination, the using unit performs its own receiving inspection. If the PEI passes this, then the RO signs for the gear and the Arrival and Assembly Operations Group (AAOG) is no longer responsible for tracking the gear.

For the period of time from approximately two weeks before the ship [or ships] pull into the port until the gear is signed for by their particular using unit, the gear is described as falling under the umbrella of the Arrival and Assembly process. Figures 9 through 13 illustrate various stages in this process, as they occurred during Exercise Native Fury in Al'Aqaba, Jordan.

Figure 9. In both images, the RO/RO Ramp of the USNS Obregon attempts to descend upon the pier at Aqaba, Jordan.

Approximately two weeks before the ship or ships pull into port, an OPP (Offload Preparation Party) which consists normally of a Warrant Officer or Staff NCO heading up a team of mechanics board the vessel or vessels and prepare the gear for the offload. During this time, they start and operationally check and inspect the equipment, order repair parts, identify any rolling stock that should be removed from the list of equipment to be offloaded, etc.

The actual offload process begins when the first principal end item (PEI) comes off of the MPF ship and ends when the last intended piece of equipment is signed for by the Supply Officer (Responsible Officer, or RO).

Figure 10. A Landing Force Support Party (LFSP) Marine conducts traffic on the POG (Port Operations Group) Pier.

Figure 11. On the left, a ship's crane hoists a HMMWV from the belly of the USNS Obregon for offloading. On the right is a close-up view of a HMMWV with an RFID tag on its grill.

Figure 12. Vehicles are lined up at the MCC, ready to be moved to the AAOE.

Figure 13. On the left, a view of the POG and the USNS Obregon are in the background. On the right, HMMWV's make their way from the MCC to the AAOE.

Figure 14. This is the AAOE Area in Al'Aqaba, Jordan.

## D. COMMAND AND CONTROL CENTERS

### 1. Executive Dashboards

The Marine Lance Corporal depicted in the image below, part of the Port Operations Group, scans a vehicle using a Radio Frequency (RF) Scanner; the underlying system is referred to as RFID (Radio Frequency Identification). This approach is similar to barcode tracking but does not repair direct close scanning of each label, facilitating rapid throughput. This system is used to quickly provide all of the information concerning USMC assets to higher headquarters and operations centers. Such information is only as valuable as those in the operations centers can make it into the effective tracking mechanism that it can be.

Figure 15. Marine Lance Corporal who is part of the Port Operations Group scans a vehicle using an RF Scanner; the system is referred to as RFID (Radio Frequency Identification), which is used to quickly provide all of the information concerning USMC assets.

In the case of the POG for an MPF Offload, research has the potential to be achieved in the arena of how to extract information given by scanning the vehicles and equipment through RFID and how to convert that data into usable XML data. That information may be used to transcribe preexisting models (as stated above) through an already built port facility in X3D Earth or an X3D field that contains the ship, the port, the operations centers (POG, AAOG, MCC, each using unit), the Movement Control Center, and the road networks in between each. Then a real-time model may be created, and entities that are offloaded and routed to the respective using units can be color-coded somehow so that each using unit who wants to know where his or her equipment is and the status of that equipment can view it easily. X3D is described in more detail in Chapter III.

Moreover, such a model may be used for training, to teach young officers and SNCO's how such an operation should look from start to finish, but more importantly, simulations can be built and potential optimization can be computed to ensure that all using units get their gear in an optimal manner, taking into account priority of unit and when all units need to be a certain readiness levels throughout the offload. This also opens up the door for artificial intelligence applications.

An important factor in relation to Maritime Prepositioning Force offloads is that such an offload is a time-sensitive event. The use of a permissive port abroad usually comes with strict timetables as to when such an offload can take place amidst ongoing commercial port operations. In many cases, MPF ships only get a couple of days to completely offload, whether they finish or not; as a result, time is of the essence.

Figure 16. This is a Maritime Prepositioning Force Ship pulling into port in support of Operation Bright Star, Nov '01.

### 2. How the COP Ties into the Overall Information Flow of the Operations Center

#### *a. Operational Center Standardization*

The future envisions the ability to visually see and track the status of such processes easily. One envisions a command operations center such as the Arrival and Assembly Operations Group (AAOG) with three major components to its visual tracking: On the left side, an Executive Dashboard, where the commander's requirements are loaded, tracked, and measured. In the middle, the COP, or Common Operating Picture, which on variable maps, shows the current up-to-date status of the process. To the right, slideshows of pertinent briefs and other miscellaneous information, to include video teleconferencing or other correspondence necessary in the AAOG or other operations center, are displayed. In short, the Executive Dashboard, COP, and Briefing Slideshow serve as a triad of Operation Center Situational Awareness (SA). In such an environment, not only does the Commander have a clear SA, but he is also enabled to operate the most effectively.

## E. RELATED WORK

### 1. Transportation Common Planning Tool (TCPT)

The Transportation Capacity Planning Tool (TCPT) is a functional logistics prototype developed for the USMC that allows transportation planners throughout all elements of the Marine Air Ground Task Force (MAGTF) to view transportation capacity in a Web-enabled environment. It also affords planners the ability to view transportation capacity over an extended planning horizon. TCPT enhances situational awareness for all levels of the organization. It was developed based on identified needs for transportation planning decision support during Operation Iraqi Freedom. TCPT was recently selected by Headquarters Marine Corps as one of six application prototypes approved for continued use by the operating forces.

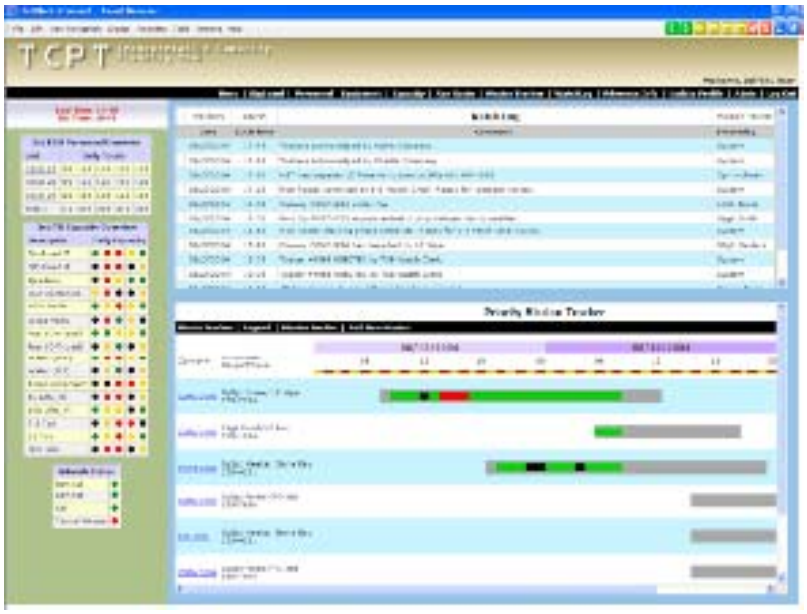



Figure 17. This GUI of TCPT displays a Priority Mission Tracker and Watch Log.

### 2. Battle Command Sustainment and Support System (BCS3)

MTS (mobile transportation system) transmits the positions of vehicles actively. It is capable of receiving multiple logistics feeds from Army and/or Marine Corps entities. The focus of this program is to receive and display, not to process or transmit. With its modems it is able to achieve position feeds of the "last tactical mile," or LTM.

With warehouse to warrior kits, users are able to tailor the view to what they want to see. Other capabilities of this system include in-transit visibility (ITV), army system PC attached chat capability, army Blue Force Tracker (BFT) antennae, and interoperability with FBCB2 and STAMIS.

MTS is primarily an Army program that simply provides a map and a receiver. This program receives feeds from either the Army's choice in vehicle tracking, MTS, or Mobile Tracking System, that sends an active signal to BCS3 letting it know where it is, or the active interrogation of RFID, or Radio Frequency Identification. Once it receives these feeds, an icon appears on this global map. The map is quite simple with only two colors, only two-dimensional, and is fairly harsh on the eyes in its color choices. The benefit of this program is its tracking of MTS vehicles; more specifically, the most impressive and integral tool in this system is the MTS. The BCS3 program itself is simple and could stand to be completely reconfigured by somebody. It does not do too much for the Marine Corps currently, which does not for some reason have MTS, perhaps due to fiscal constraints.

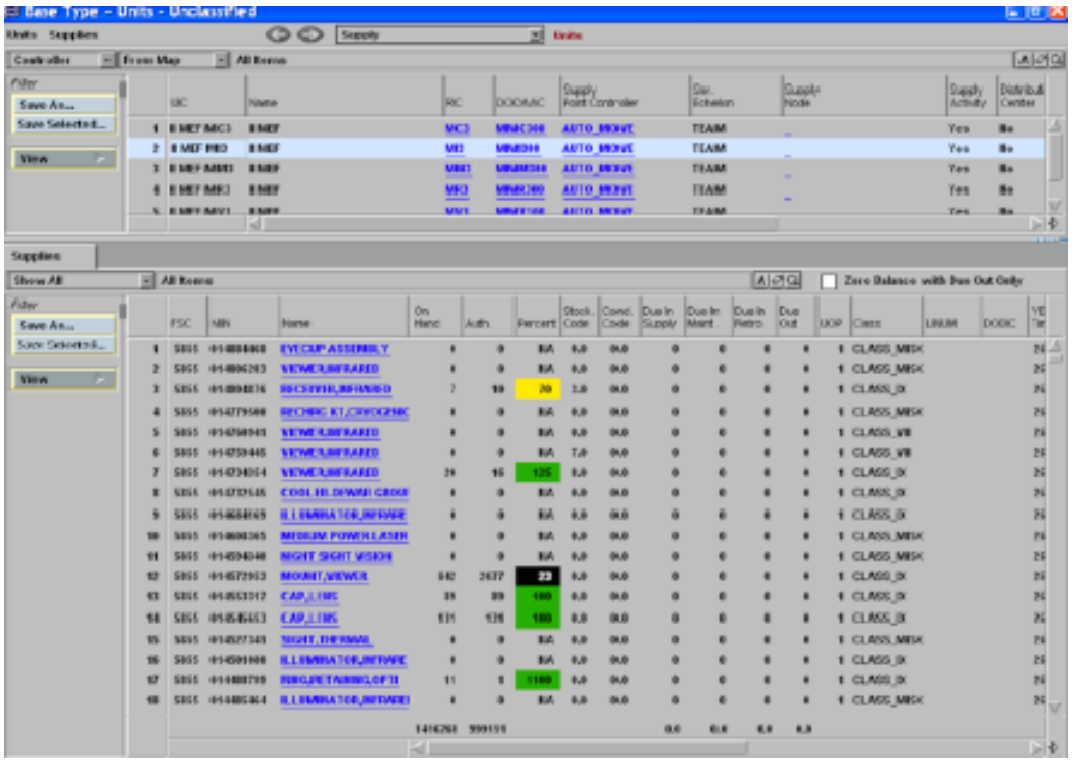


Figure 18. This screenshot of BCS3 shows a listing of entities that are being tracked by the system.

### 3. Common Logistics Command and Control System (CLC2S)

CLC2S 2.0 has an Oracle 10G Oracle Database, runs on Virtual Machines (a software implementation of a computer that executes programs like a real machine), contains a Rapid Request Process and a Rapid Request Tracking System. It interfaces with Command and Control Personal Computer (C2PC) and has an EDS (Electronic Data Systems) enhanced Combat Service Support Operations Center (CSSOC) System. This program or system advertises itself to be the answer in logistics tracking. It boasts the ability to perform administrative tracking of personnel, maintenance tracking of equipment, an ability to complete and communicate rapid requests, and perform engineer tracking. It does organize information needed to track personnel and equipment in a logical manner. The assignment of red, amber or yellow, and green to the status is arbitrary, and in contrast, the commander usually determines the measure of urgency. Moreover, the way in which the program was written does not allow for the ability to move from a lower-level unit to a higher-level unit once a user drills down to locate a specific person or piece of gear.

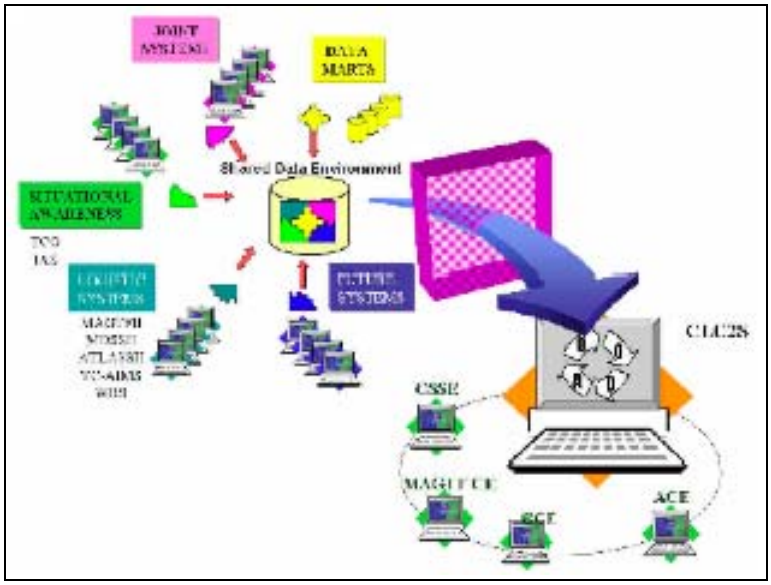



Figure 19. Showing the information flow from a shared data environment to the common logistics program, CLC2S serves as a universal dashboard linking logistics status to readiness hues.

### 4. Expeditionary Decision Support System (EDSS)

EDSS is an Office of Naval Research (ONR) expeditionary warfare-planning segment for Global Command and Control System-Maritime (GCCS-M). This tactical decision aid incorporates operating features of and shares a common database with the existing GCCS-M Mine Warfare Environmental Decision Aids Library (MEDAL) segment allowing extensive data sharing between mine warfare and amphibious warfare planners. Currently under experimental fleet use by deploying Amphibious Ready Groups (ARG) and their staffs, this program is currently deployed on a majority of amphibious ships involved in Operation Iraqi Freedom.

Glenn Palmer has been using this program to aid in military planning for the past seven years; he currently works at COMNAVBEACHGRU 2. He reports that "mission planning gurus" have utilized this system to simulate ships, the sea echelon area, and serial assignment tables in conjunction with a C2PC injector. The C2PC feeds into it, and it inputs ICODES data, the number of craft, serial assignment tables, serialized waves, on call waves, and several other pertinent parameters to aid in providing a planning tool for large-scale operations. The routes are timing programmed; given H-hour, this program projects the timeline of the operation. This program shows movement and simulation. It is primarily a planning tool; though not so much intended as a tracking tool, yet it has standard tracking capability. It writes operational tasks and puts them into outgoing messages directly. MDSSII Data has 45 fields, but this program only uses nine or ten fields, including description, UIC, category code, etc. Figure 20 shows the capabilities of this system in illustrating planning for military operations and operational movements.

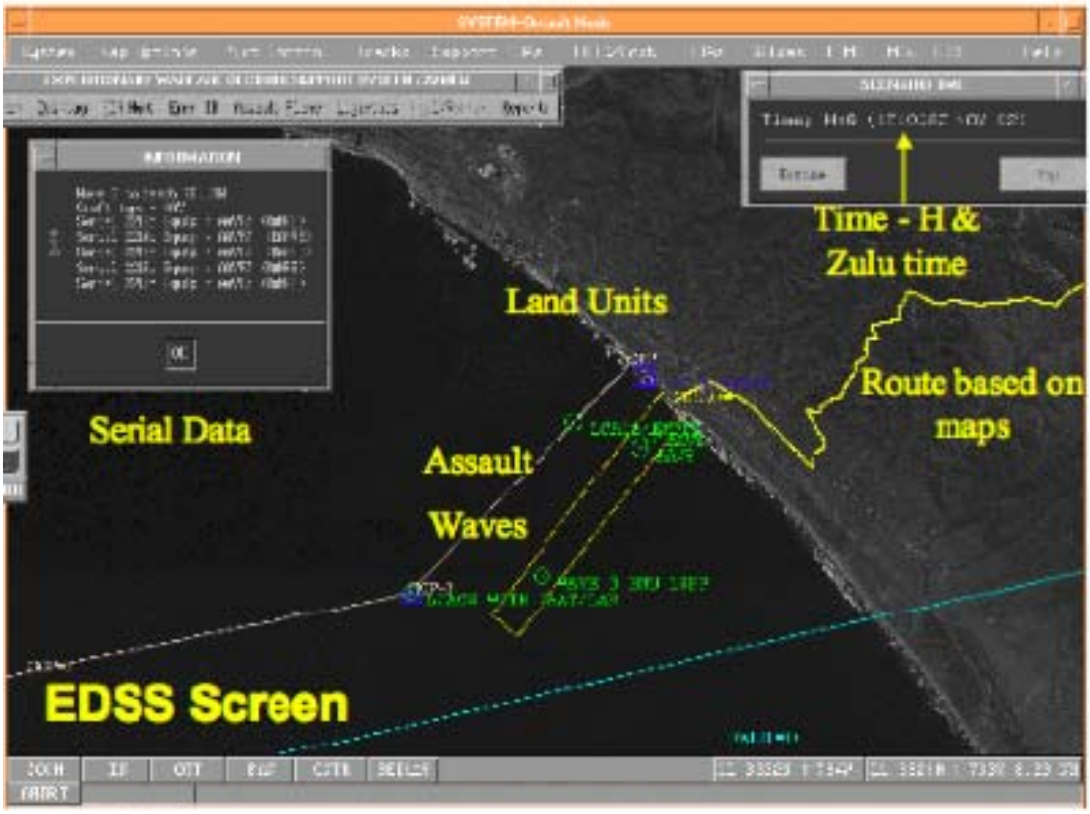



Figure 20. This GIU shows the capabilities of EDSS, that of illustrating planning for military operations.

## 5. Interoperability Tools

### *a. Blue Force Tracker (BFT)*

Blue Force Tracker is a United States military system that helps to provide commanders with information about their forces. In military symbolism, the color blue is used for friendly force elements, red is used for enemies, and green is used for neutral forces. The Blue Force Tracking system consists of a computer, satellite antenna, and Global Positioning System receiver. The system displays the location of the host vehicle on the computer's terrain-map display along with other platforms in their respective locations. BFT can also be used to send and receive text messages and Blue Force Tracking has a mechanism for reporting the locations of enemy forces and other battlefield conditions. Users include the United States Army, the United States Marine

Corps, the United States Air Force, and the United Kingdom, but the Marines primarily use the Mobile Data Automated Communications System to provide situational awareness. To ensure that Marine force locations will be visible in U.S. and British Army command centers, the Marines have installed more than 200 BFT systems as well. Work has begun on plans to reach the level of nearly 40,000 tracking systems in the Army within four years.

A Blue Force Tracking technology is system called Force XXI Battle Command Brigade and below, or FBCB2. The system continually transmits their actual locations over the FBCB2 network. It then monitors the location and progress of friendly forces and sends those specific coordinates to a central location called the Army tactical operations center. There the data is consolidated into a common picture and sent back out to units. The system also allows users to input or update operational graphics (i.e., obstacles, engineer reconnaissance on the road, etc). Once uploaded, it can either be mailed back to higher headquarters or 'mailed' to other subscribers of that user's list. An additional capability comes from the route-planning tools. By inputting grid coordinates, the BFT becomes both the map and compass for motorized units. With proximity warnings enabled, the vehicle crew is made aware as they approach critical or turn points. Figure 21 shows the networking flexibility and capability of Blue Force Tracker. This capability allows for ambitious distributed operations.

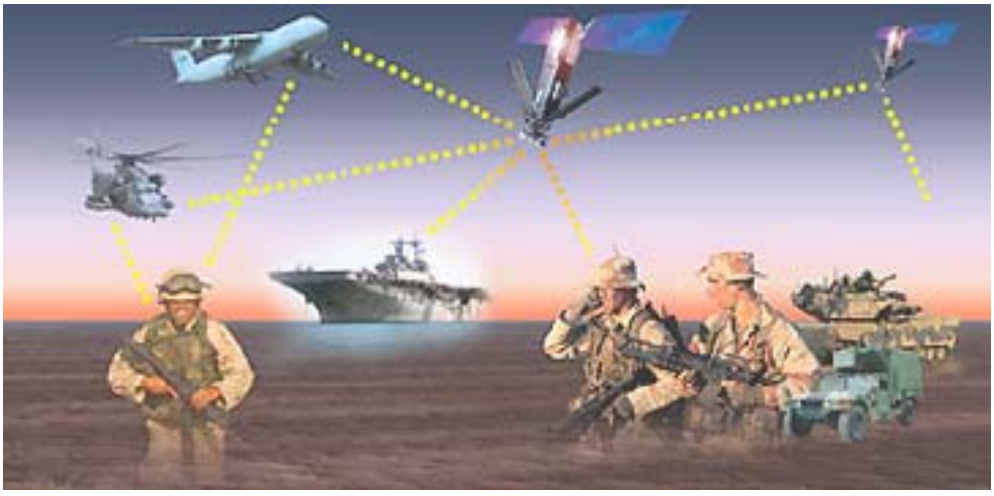


Figure 21. This is an image of Blue Force Tracker networking capability.

### *b. RFID Data Tracking, Passive and Active Interrogation*

Radio-frequency identification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. An RFID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification using radio waves. Some tags can be read from several meters away and beyond the line of sight of the reader.

Most RFID tags contain at least two parts. One is an integrated circuit for storing and processing information, modulating and demodulating an RF signal, and other specialized functions. The second is an antenna for receiving and transmitting the signal. Chip-less RFID allows for discrete identification of tags without an integrated circuit, thereby allowing tags to be printed directly onto assets at a lower cost than traditional tags.

Today, RFID used is in enterprise supply chain management to improve the efficiency of inventory tracking and management. However, growth and adoption in the enterprise supply chain market is limited because current commercial technology does not link the indoor tracking to the overall end-to-end supply chain visibility. Coupled with fair cost-sharing mechanisms, rational motives and justified returns from RFID technology investments are the key ingredients to achieve long-term and sustainable RFID technology adoption. Figure 22 shows an example of an RFIT tag on the front of a HMMWV. This placement allows for more effective interrogation by interrogators.

Figure 22. An RFID Tag on a HMMWV allows rapid scanning at a distance, facilitating inventory control during tactical exercises or operations.

### *c. Signpost*

Signpost was fielded this summer in support of Exercise Native Fury in Al'Aqaba, Jordan, July 2008. This system allows for localized tracking of gear and equipment as it traverses through the offload. Essentially, it acts as a set of gates through which and over which vehicles pass, actively interrogating their RFID tags and reporting this information via a meshed network to higher headquarters. Thus far this system has proven itself mostly successful. Figure 23 shows the employment of the signpost system in the field, here at Exercise Native Fury in Al'Aqaba, Jordan. The image on the left displays one half of the signpost system, while the image on the right shows the "gates" through which RFID tagged vehicles pass.

Figure 23. These images show Signpost gear being tested in Al'Aqaba, Jordan.

## F. SUMMARY

In conclusion, each of these related technologies provide different capability to enhancing the logistics capability of the force. However, open source, readily available simulation tools such as Viskit and X3D, described in detail in Chapter III, simulate the logistics process as a whole and aid in better, more coherent command and control at the operational level.

# III. SIMULATION TOOLS

## A. INTRODUCTION

This chapter focuses on the simulation tools of choice to tackle the problem outlined in Chapter I, the problem of simulating the MPF offload process and capturing it in a 3D environment.

## B. OVERALL METHODOLOGY

The following tasks characterize the overall methodology used to model and simulate MPF offloads.

### 1. Capture the Data

This step can be achieved using either a front-based or end-based approach. A front-based approach includes capturing all relevant data for each of the scenarios and converting it into XML. In an end-based approach, what is desired to be accomplished is observed, and only the data pertinent to those goals is converted, rather than all of it. The upside to the latter tactic is that less data is converted, but there may be applications for which the creator might not yet realize that are beneficial to the overall effort.

### 2. Analyze and Determine Utility/Possible Applications

At the very least, the data can be transformed into HTML and made available over the appropriate channels, be it the Internet or SIPRNET. The data is best suited for simulations in a planning environment, and working with interoperability tools to achieve real-time tracking to provide a Common Operating Picture, COP.

### 3. Applying Programs, Simulations, Analysis, Optimization, Artificial Intelligence, etc., to Achieve Results

The Logistics Operations Center must monitor Vehicle Status Reports, Personnel Accountability Reports, Levels of Classes of Supply, where all recovery and emergency vehicles, personnel, and equipment are, what priority of support is given to whom and when, all Support Requests, the statuses of all convoys that are out, etc. The Logistics Operations Center is nothing short of one big brain, one large nerve center in which

inserting a heavy dose of modeling and simulation to all aspects of this operations center does nothing short of adding coherence to a tedious palace of chaos and constant potential confusion. Here more than most places the benefits of artificial intelligence can be felt, for so many decisions rely on priority that oftentimes a computer does a much better job than humans do. Also, so many things in such an operations center are repetitive that they are silently screaming the need for computerized help, and not in the form of many systems and many programs that overlap and cannot speak to one another. Getting all of the data into XML and using that as a platform to do everything else can ensure a much more highly operable operations center.

#### 4. Technical Approach

Once all of the pertinent data is converted into a readable form of XML, it can be harnessed by Savage tools, to include Savage Studio, SMAL, X3D Earth, X3D Edit, Viskit, and Simkit, open source applications that allow for great durability, flexibility, and usability. These applications are discussed in great detail throughout this chapter.

## C. PROGRAMMING CONSTRUCTS

### 1. Java

Java is an object-oriented programming language developed by Sun Microsystems. Java was designed to be platform independent so a developer could write a program once and run it on any arbitrary set of computer hardware. Java is used extensively at NPS for that reason and because most Java development tools are free. Java is primarily used for M&S because of its platform independent design, its multi-threaded capability, and the multitude of available related open-source code.

### 2. JAXB

JAXB is an open source API created by Sun Microsystems. It provides a convenient way to bind XML schemas to Java source code representations. JAXB makes it easy for developers to incorporate XML data and processing into applications. As part of this process, XML documents are either marshaled to Java classes or un-marshaled into a JDOM tree for use by the program.

### 3. Document Object Model (DOM)

The Document Object Model (DOM) is a platform and language-neutral interface and World Wide Web Consortium specification that allows programs and scripts to dynamically access and update the content, structure, and style of documents. Sun Microsystems has implemented the DOM interface a component API of JAXB in the ord.w3c.dom Package. It allows programmers to create, modify, access, and write XML documents using the Java programming language. Additional information is available at Sun's website at http://java.sun.com.

### 4. Extensible Markup Language (XML)

Extensible Markup Language (XML) is a general purpose, text-based markup language developed by the World Wide Web Consortium (W3C) as a subset of Standard Generalized Markup Languages (SGML). Like all markup languages, it was created as a protocol for structuring data. It is not a programming language but it makes it easy for a computer to generate the data, read data, and ensure that the data structure is unambiguous. XML is east to create and process and designed to be platform independent and shared across the Internet. Other characteristics of XML include human readability, extensible, verbose, modular, and license free. More information can be found at www.w3.org/XML/1999/XML-in-10-points.

## D. SIMULATION AND PROGRAMMING CONSIDERATIONS

### 1. Discrete Event Simulation (DES) Modeling Characteristics

Models are created to study complex dynamic systems and examine their performance, reliability, or other properties to improve either their initial design or operation. Simulation is the means of executing these models to mimic the behavior of actual systems. Simulations employ many repetitive runs to obtain relevant statistical output for insight into the actual operation of the modeled system without real-world testing that is often impractical and costly. In general, if a model uses an equation to define a characteristic, then a simulation is the behavior or trajectory of that function over time.

There are two defining characteristics when creating a model. First, fidelity, measures the level to which the model reflects the characteristics of the real system, like how similar it is in shape or dimension, physical characteristics or constraints, or performance. The second, abstractness, measures the lack of level of detail. This is required not simply because it is impossible to capture every detail of a real-world system that may or may not be known, but also because it allows for generality. Generality is beneficial, since it likely allows for the design and analysis of multiple models simply by changing parameters.

The ideal model has both high abstraction and high fidelity. Unfortunately, these are competing requirements and therefore a compromise must be reached, one that usually depends upon the needs of the modeler. The design of this particular model is highly abstract and repeatable, while the test cases developed are highly specific and realistic.

#### *a. Simulation Approaches for Handling Time*

There are two broad types of simulation modeling primarily characterized by how they handle the passage of time. The first is Continuous Systems Simulation (CSS). CSS is creating a model that can be represented by differential or difference equations. In essence, it breaks the time domain into quantized chunks of small (usually the same) size. The second is DES, Discrete Event Simulation, and is also the focus of this thesis. It differs from CSS because it divides the time domain by events. According to Arnold Buss of NPS, DES has three main worldviews: Event-Scheduling, Process Interaction, and Activity Scanning. The Event-Scheduling approach is based on the use of event lists to organize future events. This is the world-view utilized in Simkit, and therefore is utilized in this thesis.

#### *b. Methodology*

Events are actions defined by the modeler to represent basic functionality of the simulation. They represent changes in state that typically takes some amount of time to occur, such as an object arriving to the queue or a server completing a job. The

event list is simply a container that holds the list of events that are scheduled to happen and the time at which they will happen. Buss describes the event list as the following:

> The Event List amounts to a "to do" list for the simulated world. At any simulated time epoch it is simply a list of what is scheduled to occur and when. Each item of the list corresponds to an event that contains information about which event is to occur and when it is to occur.

The scheduling and manipulation of the event list is the engine driving a DES. Every action that comprises the model will be scheduled on the event list. Time advances only in intervals defined by the time difference between the current event time and the event on the event list with the smallest time duration. This process continues throughout the duration of the simulation, that is each event is drawn off the event list one at a time ordered by the time the event is scheduled to occur, until the event list is empty or an event is scheduled that explicitly stops the simulation. Note that it is possible for two events to be scheduled for exactly the same time and therefore it is necessary to implement an order of precedence procedure in the event list.

#### *c. Notation*

An event graph is a structured, formal representation of a DES model. Schruben defined event graph notation in work in 1992. This notation is minimalist in that it uses only those entities that are required, but in doing so adds a level of abstractness not seen in other DES world views, such as Process-Interaction (Buss 2001). The advantages of adhering to this notation are that virtually any model can be constructed with it and the modeler can spend more time on model creation vice paradigm constructs. Using the following notation conventions, the modeler can graphically depict all logic and behavior contained in the model.

Additionally, this event graph provides no method to begin the initial event A and therefore it must be initialized by a foreign event either programmatically or through a listener pattern. Normally every event graph contains a Run event from which other events are propagated and to reset all state variables when a simulation run is completed.

Two optional components of scheduling edges that greatly extend the functionality of event graphs are edge conditions and time delays. Edge conditions are conditional expressions defined by the modeler that prevents the edge from being invoked until said condition is true. Edge conditions are represented by logic functions above the wavy line in the middle of the scheduling edge. Time delays, represented by a (t) located at the start of the scheduling edge, control exactly when from execution of event A that event B is to be scheduled. Therefore, the event graph depicts that once event A is scheduled, event B will be scheduled (t) amount of time later if expression (*i*) is true.

Two final and necessary components of complex event graphs are parameters and state variables. Parameters area variables defined at the start of each simulation run and represent constructs such as total number of servers or number of targets to be created. State variables on the other hand are variables designed to change throughout the simulation run. As the name suggests, state variables are updated to reflect the changed state of the model such as the number of people in the queue at a particular time. Using this notation, it is possible to construct models of limitless complexity. Depicted is a more complex model of a transfer line process where a component is passed from one server to another and is finished only when processed by all servers. In this model, Q and S represent state variables and (*i*) represent a passed parameter. Figure 26 shows an event graph for the Transfer Line process.

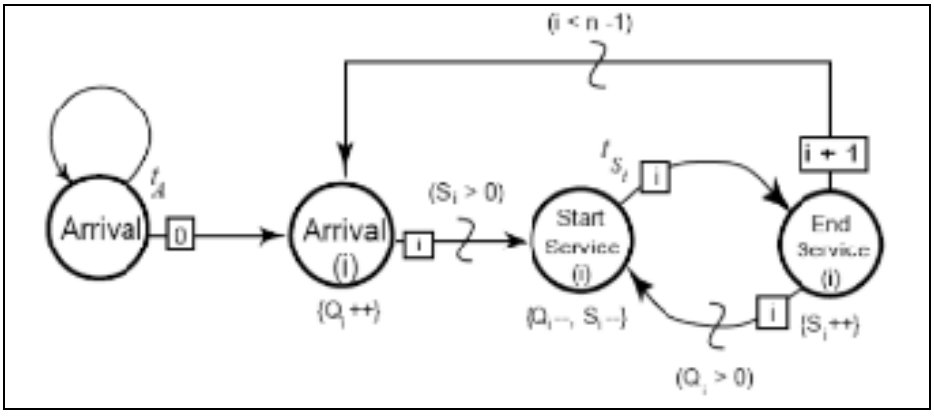



Figure 24. This is a complex event graph of a Transfer Line Process (Buss 2001).

### 2. Simkit

Simkit is an open source Java API written by Professor Arnold Buss of Naval Postgraduate School. It was designed to enable the creation of DES using event graph methodology. In short, a simulation can be created programmatically using Simkit because it provides the base framework for controlling the simulation, namely control and maintenance of the Event List. This frees the modeler to work directly on implementing the conceptual event graph model. Simkit also provides other helper classes necessary to create simulations. These include random variate generators that produce random numbers in the required distributions, and classes that facilitate movement and detection among others. Simkit is the foundation upon which Diskit and Viskit are built. It is possible to create complex DES using Simkit and the following website lists the NPS Master's Thesis work that has been completed using Simkit (http://diana.nps.edu/~ahbuss/#Students accessed on March 2007).

### 3. Diskit

Diskit is another open source Java API that extends the functionality of Simkit. It was created for two primary reasons. First, there was a need to extend the movement and detection capabilities of Simkit to 3D. This is because 2D usually doesn't provide the level of fidelity required for a model that simulates movement. Secondly, Diskit provides classes that implement the Distributed Interactive Simulation (DIS) protocol. DIS allows for transmitting the state of a simulation over a network. DIS coupled with the extension to a 3D environment enable the visualization of the simulation as a 3D virtual environment.

### 4. Viskit

Viskit is an open source program in development at NPS written in the Java programming language. Viskit was created to provide a graphical user interface (GUI) for creating simulations using Simkit. Typically, creating complex simulations is programmatically intensive. A modeler usually needs an extensive knowledge of a programming language and the associated APIs that enable the simulation. This is no different for Simkit and Diskit, and is exactly why Viskit was developed. By reducing the amount of programming expertise required, Viskit has made simulation more

accessible to non-programmers. Viskit uses a tabbed window with four tabs. The first provides a visual interface that allows for easily creating, modifying, and saving event graphs called the event graph editor. Figure 25 provides an example of a simple event graph in Viskit's event graph editor. It demonstrates that event graphs produced in Viskit faithfully adhere to the event graph methodology presented earlier. This ensures that if event graph methodology is understood, Viskit can represent it and others who have no familiarity with Simkit or Diskit can understand it. Additionally, because the source code is automatically produced by Viskit, it allows for more complex event graphs to be created without being increasingly encumbered with programming complexity that might quickly become unmanageable.

Viskit stores event graph models as XML documents. Below a simple event graph is depicted, the Arrival Process, and the next figure is the XML representation of it in Viskit. JAXB enables the XML structures used by Viskit to store event graphs to be transformed into executable Java source that can then be utilized by Simkit and Diskit. This allows developers to create DES models quickly using only standard event graph notation and methodology without having to master the Java programming language. XML is used in Viskit as the format for saving Event Graphs and Assemblies. Figure 25 shows a simple arrival process, while Figure 26 shows an arrival event graph in the Viskit program.

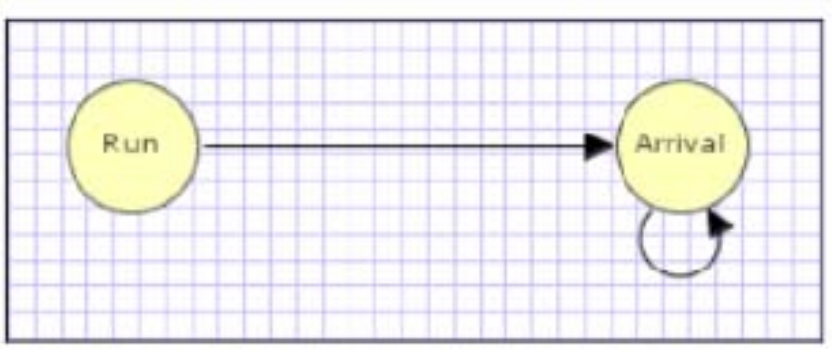



Figure 25. The above figure shows the arrival process in Viskit.

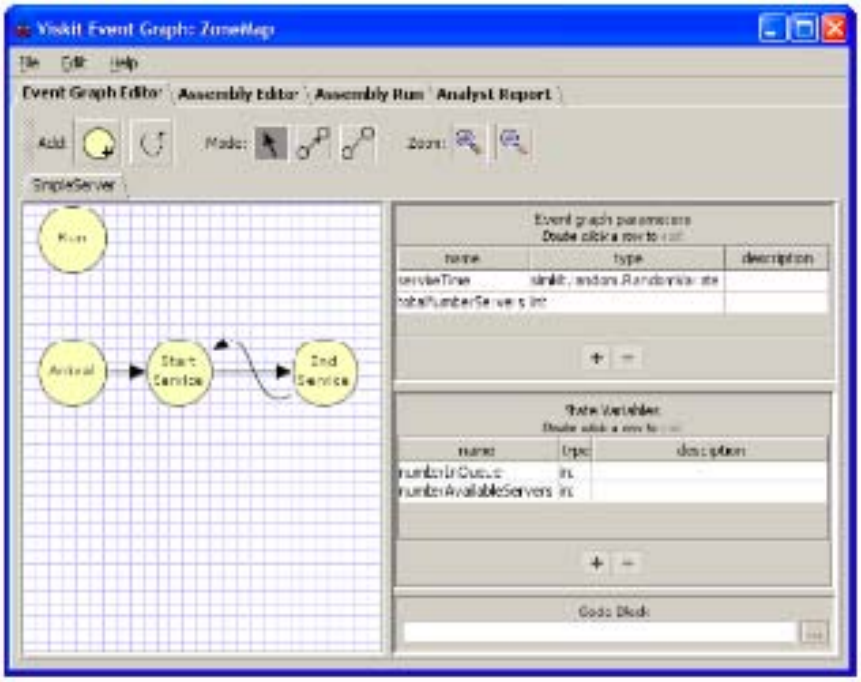



Figure 26. This is a basic event graph depicted in Viskit.

Creation of the event graph representation of a model alone does not create a run able discrete-event simulation. Viskit provides a means to create, modify, and save simulations using event graphs in a panel called the Assembly editor located in the second tab. Event graphs that were created in the event graph editor (or any event graph created that extends Simkit's SimEntityBase) show up on the left panel and are drag and dropped to the right workspace. They are then connected using listener patterns.

Finally, statistics-counting objects are listed in the lower left panel and drag and dropped to the workspace on the right as required where they are connected to the event graphs with PropertyChangeListeners. This will produce applicable and repeatable statistics as required of the simulation.

The Assembly Run is the third tab and provides a location to run the simulation, illustrated in Figure 27. In the Assembly Run panel after a run of an exemplar simulation, on the left are controls to modify run parameters of the simulation such as length of time to run and how many times to run the simulation. The top right of the

Assembly Run panel contains the text output of the simulation that can be inspected after each run. The bottom right of the panel provides an error report generated during the run.

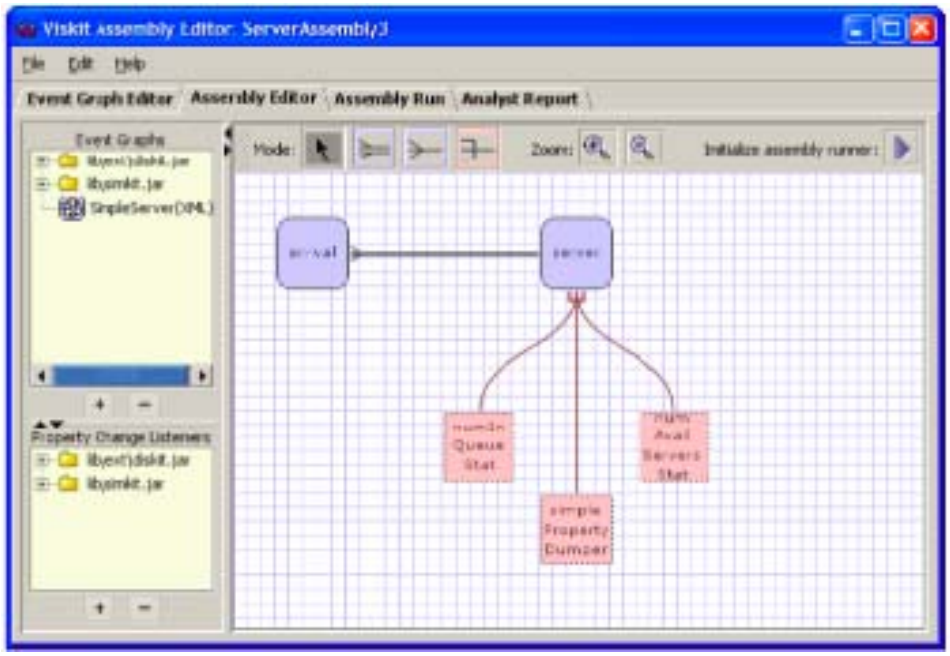



Figure 27. This is an assembly editor example in Viskit.

## E. DES AUTHORING – CREATING A SIMULATION WITH VISKIT

### 1. Simkit/Diskit API Library Inheritance Structure and Use in Viskit

Computer programs are complex constructs that if coded in a single container would extend many lines and pages. Java and indeed most programming languages provide mechanisms to organize and reuse code as much as possible. Viskit simulation architecture utilizes all those inherent to Java, but one is of particular interest in Viskit called inheritance. Inheritance allows for many implementing objects to contain all the inherited characteristics of the super class while allowing for the addition of a new functionality in the current class. Additionally, the current class could then be used as the super class for another class, etc, etc. In effect, inheritance provides the ability to create multiple entities that are primarily equivalent, yet has unique functionality. The concept of inheritance is especially important to entity creation in Viskit. As entities become

more complex thru the process of implementation of features, the event graphs can become overwhelmingly complex. A fully functioning Viskit entity is difficult to decipher, modify, or test for desired behavior.

Viskit allows for the use of inheritance and it has been demonstrated that its judicious use is essential to producing a maintainable model and is a 'best practice' and simply a good design pattern that should be followed. Viskit provides this ability thru the event graph settings dialog box that allows for specifying which class to extend. It is important to note that to be used in Viskit, at some point one of the super classes must be SimEntityBase. The benefits to adhering to this best practice include event graphs that are more readable, focused on implementing only what is different from the super class, and easier to debug.

#### *a. SimEntityBase*

SimEntityBase is the fundamental component of Simkit simulations. Recall that there are just two constructs of event graphs: the event and the scheduling edge. SimEntityBase is the class that controls interactions with the event list. Each event on an event graph is placed or removed from the event list according to its scheduling edge. Every event graph in a simulation or one of its super classes must inherit from SimEntityBase at some point otherwise it cannot interact with the event list.

#### *b. Mover3D*

Mover3D is a Java interface that ensures implementing classes meets the minimum requirements for a 3D mover in Diskit. It is essential that all movers in Simkit requiring interactions such as detection and its inverse, un-detection, implement Mover3D. This is because of how the sensor classes are constructed since they fire 'doDetection' and 'doUndetection' events as specified by Mover3D. By convention, implementing classes are named with Mover3D appended such as DISMover3D.

#### *c. DISMover3D*

DISMover3D implements the Mover3D interface and extends SimEntityBase, thus it provides the minimum constructs for a Simkit simulation as well

as ensuring it will interact properly with the sensor library. Additionally, it provides all the functionality required for a moving entity along with exposing that entity to the DIS protocol. Essentially, DISMover3D provides all the functionality required of a simple 3D mover that can detect other object and output its state as DIS packets across a network.

### 2. Event Graph Editor – Creating a Model

Event graphs define a DES and control the behavior of the entities and their relationships to other entities. Producing a productive simulation requires creating event graphs that encompass behaviors of sufficient fidelity while maintaining some requisite amount of generality.

#### *a. Event Graph Parameters*

In Simkit DES methodology, event-graph parameters are variables that are set at simulation run time and do not change during the run. The exact value of parameters must be entered in the Assembly Panel prior to the start of the simulation or an error will occur. Parameters also represent performance characteristics of a model. These must be available as changeable parameters to maintain a sufficient level of abstractness to allow for multiple simulation runs without modifying hard-coded values.

#### *b. State Variables*

A state variable is a mathematical variable that defines an important aspect of the system. State variables change throughout the simulation and that change is called the state trajectory. The state trajectory is the graph of change in a state variable over time or "evolution of the model in time." (Buss 2000) Each state trajectory is piecewise constant and therefore only changes at events. In a typical non-moving entity simulation, most events represented on event graphs contain state changes. This is not true for tactical models where decisions and behaviors by an entity are captured. In Viskit, state variables are entered and listed on the Event Graph Editor panel.

#### *c. Events*

Events are one of the two fundamental components of event-graph methodology (the other being the scheduling edge). They are graphically depicted as

circles on the Event Graph Editor panel. When creating an event graph, empty events are placed on the graph and then information about that event is entered into the Event Inspector that is accessed by double-clicking that event.

The Event Inspector is used to define events and consists of four main components: Event Arguments, Local Variables, Code Block, and State Transitions. The Event Inspector of a Start Moving event and its main components are an example. Beginning with the Event Arguments section, the functions of the sections are explained. Arguments of events are incoming values and are directly analogous to the signature of a Java method. The composition and position of arguments determine the signature. Any call to that event from a waitDelay() method must be spelled correctly and have the exact same signature or nothing will happen. If an event has arguments then any attached upstream scheduling edges must provide the value of that argument.

All sections on the Event Inspector have plus and minus buttons used to add or remove elements. Clicking the plus button adds an empty argument and double clicking it brings up the Event Argument dialog box. The event argument dialog box is used to define the argument's name and type. The Local Variables section provides a location to define variables whose scope is limited to that event. They can be defined for any function, but are typically used to supply values to scheduling edges without referencing the original object. Clicking the appropriate plus button in the local variables section add local variables. Double click the new entry to define a new local variable in the resulting Local variables dialog box that appears. The new variable is defined by its name, type, and initial value.

The Code Block section in the Event Inspector is a free-form code entry area. It is used to enter any required code whose function cannot be performed by one of the other sections. The code must adhere to Java language programming syntax and rules. Unlike previous sections, code is entered directly on the provided line or if more space is needed, in the box accessed thru the ellipse notation to the right. The most common functions for code in the Code Block are print statements used for debug purposes and helper classes for data collection.

The final section is for state transitions. Similar to previous sections, users can add and remove state transition entries with the plus and minus buttons. Double clicking an entry brings up the State Transition dialog box. In the dialog box, select the appropriate state variable that needs modification and then give it a new value directly or through a function. Note that only state variables previously entered in the Event Graph Editor are available for change.

All event graphs are basically linear programs that move sequentially from one event to the next thru scheduling edges. Most event graphs start with a Run event that initializes all state variables and resets then upon multiple simulation runs. Following the Run event, the system moves systematically to the next event as directed by the scheduling edges.

#### *d. Scheduling Edges*

The second main component in event graph methodology is the scheduling edge. Scheduling edges connect two events together, and as their name implies, serves as a method to transition from one event to the next. In Simkit, edges are implemented by waitDelay() methods. The waitDelay() has four components: the scheduled event name as a String, the time delay from completion for scheduling (source) event to scheduled (target) event, the priority of events if there are two or more on the event list scheduled at the exact same time, and the target event parameters.

In Viskit, the waitDelay() method and therefore the scheduling edge are depicted by an arc ending in an arrow from the source event to the target event in the Event Graph Editor. The edge is defined in the Edge Inspector dialog box accessed by double clicking the graphical edge. One additional property of the scheduling edge is the conditional expression. It lists the conditions that are required to be met prior to the target event being scheduled. This determines if the edge will schedule the target event.

### 3. Assembly Editor – Creating a Simulation

Viskit defines a construct called the assembly that it uses to create the simulation. An assembly is constructed in the Assembly Editor panel of Viskit. An assembly is a collection of event graphs and the connections between them called SimEventListeners

(SEL's). The relationship between event graphs and the information passed between them via the SEL defines the foundation of the simulation.

#### *a. Scenario Manager*

The Scenario Manager is a required element of simulations utilizing Diskit components. It provides all the functionality required to implement movement and detection as well as the DIS protocol. This allows the modeler to create movers with sensors easily by connecting the Scenario Manager and the mover event graph with a SimEventListener (displayed as a line with a small cup at the end symbolically representing an ear listening to the event graph). Additionally, implementation of the DIS protocol enables the simulation to publish DIS packets to a network enabling distributed simulation and graphics. Parameters are accessed through a dialog box called the Event Graph Inspector by double clicking the event graph representation in the Assembly Editor panel.

#### *b. SimEntity*

Once even graph models of SimEntities and objects have been created either in the Event Graph Editor or as native Simkit Java classes, they then can appear in the event graphs section of the Assembly Editor. If they appear in this list then they meet the requirements of Viskit and can be used in the assembly o create a simulation. To use the event graph, simply drag and drop it onto the assembly to create an instance of it as represented by a SimEntity Node. Scenario Manager is not an event graph, but can be accessed and used in the assembly since it is a Java class that extends SimEntityBase.

#### *c. Parameter Entry*

As discussed previously, event graph parameters are values that will not change throughout the simulation and are set at runtime from entries in the SimEntity nodes of an assembly. Parameters are accessed via the Event Graph Inspector dialog box by double click. Parameters can be simple numerical values like integers or doubles, or can be any other Java function that returns an object or value.

### *d. SimEventListener*

Lines connecting SimEntity nodes in the Assembly Editor represent Simkit constructs called SimEventListeners. SimEventListeners connect two SimEntity nodes together and allow them to share information between them. To connect nodes via a SimEventListener connection, each event graph must contain an identical event (same name and signature). The source event fires then as a result the target event is fired. This has the effect of one event listening to the other event, hence the name SimEventListener. SimEventListener connections are very beneficial to the Simkit methodology of creating simulations because they allow for passing of information between event graphs. This enables the componentization or breaking up of complex event graphs into small chunks of functionality that allow for extensive re-use and simpler debugging. The concept benefits are therefore similar to the use of inheritance.

### *e. Property Change Listener (PCL)*

The Property Change Listener (PCL) is similar to the SimEventListener in that it is a construct that can be programmed to listen to a SimEntity node. Unlike the SimEventListener connections that listen for events, PCL connections listen for changes in state variables. By convention in Simkit, every time a state variable changes, a method called firePropertyChange() is initiated that has the effect of broadcasting that change to the simulation environment.

PCL connections are created to listen for specific state variable changes then perform preset operations with them. Typically, these operations are for the collection, calculation and display of statistics. Simkit has a number of built-in data collection and analysis objects that can easily be incorporated into a simulation. In the bottom left display resides the expanded list of included PCL connections, and the assembly depicts one PCL connection node (colored pink). New data collection objects that implement PCL connections can be created and easily incorporated into a Viskit simulation via the plus button. To incorporate a PCL into a simulation, a PCL is selected from the list (or created in Java if one in the list does not fill all requirements) based on the type of property being collected such as an integer or a collection and the property's state as a function of time.

Once the appropriate PCL is selected, simply drag and drop it onto the assembly and connect it to the SimEntity node with the correct state variable. Finally, select the appropriate state variable from the list accessed by double clicking the connector.

### 4. Statistical Results

The output of a stochastic simulation is typically statistics of static or time-varying nature. Each run of a simulation produces another set of statistics called a repetition. A number of repetitions are produced which are used to generate confidence intervals. Confidence intervals are then used to analyze the model for expected behavior and to generate logical inferences from unexpected behavior.

The Assembly Run panel is the location where each repetition and calculated confidence intervals are viewed. The Assembly Run panel is the location where the simulation is initiated and its control settings adjusted.

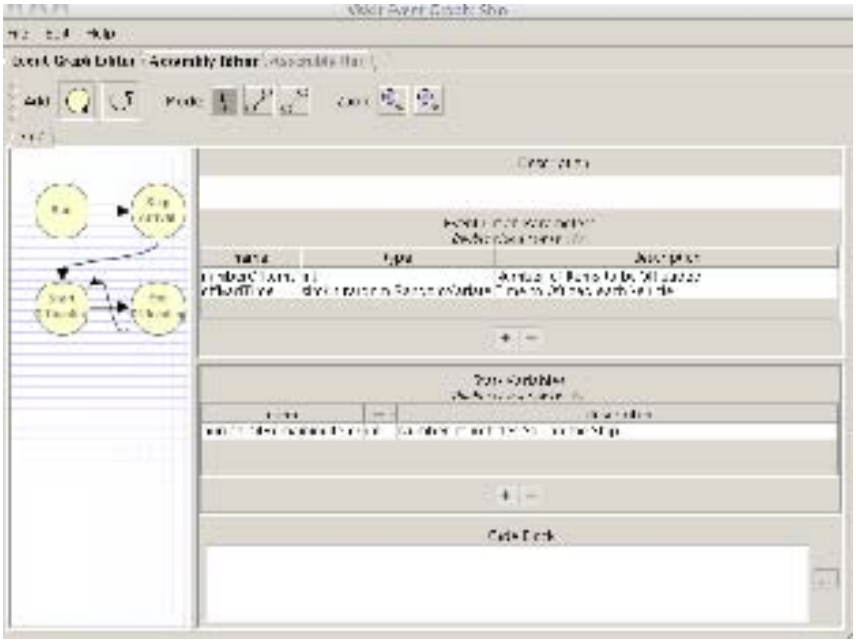


Figure 28. This is a Viskit screenshot.

## F. X3D

X3D is the ISO Standard (International Organization for Standardization) XML-based file format for representing 3D computer graphics, the successor to the Virtual Reality Modeling Language (VRML). X3D features extensions to VRML (e.g., Humanoid Animation (HANIM), NURBS (Non-uniform Rational B-Spline), GeoVRML, and so on), the ability to encode the scene using an XML syntax as well as the Open Inventor-like syntax of VRML97, and enhanced application programmer interfaces (API's).

## G. SUMMARY

In conclusion, the java-based, web-enabled, and XML-intensive open source system of programs that this suite of programs envelopes enables users to create scenarios, in this case, of a logistics system, and adds the flexibility, repeatability, and usability that today's system technologies need.

# IV. REPRESENTING OPERATIONAL LOGISTICS SCENARIOS, PUTTING IT ALL TOGETHER

## A. INTRODUCTION

The elements of extreme programming are used in the application of a discrete event simulation to this logistics process. The end state is defined, and a base scenario begins the journey to simulating the process. As each scenario is built upon its predecessor, it becomes more complex and closer to the truth of the process on the ground.

## B. EVENT GRAPHS AND ASSEMBLIES

Each event graph describes a simple process that occurs inside the larger logistical processes. These event graphs are connected into assemblies, where the full expressions of the processes are evidenced.

### 1. Assemblies in Viskit: MPF Offload

#### *a. Scenario 1: Base Scenario*

In the first scenario, most of the time distributions are set to exponential distributions for simplicity's sake. Also, to make the first scenario a baseline scenario, it is assumed that all "entities" or "items" – Principle End Items (PEI's)—flow seamlessly through the MPF Offload Process. Only one ship is offloaded, a JLTI Process is performed, the PEI's are each accounted for by the POG, gear is associated/disassociated then moved to the Movement Control Center, where each PEI moves in the same order from there to one sole AAOE area where it is received. Table 3 illustrates the processes each item must go through as well as descriptions of each process and the time distributions used to approximate the length of time it takes to perform each task. Figures 29 through 35 show the event graphs for each step in the offload process, each event housing its own event graph. As an example, Figure 30 describes the Joint Limited Technical Inspection Process through which all Principle End Items (PEI's) must pass. The limiting factors in this process are the number of mechanics available at any given time, as well as the length of time it takes for each mechanic to End the Service

performed, that being a mechanical inspection of each vehicle or other PEI. All of these event graphs are combined in the Assembly Editor annotated by Figure 36, where their relationships to each other are determined.

| ***Process*** | ***Description*** | ***Constraint*** | ***Time Distribution*** |
|---|---|---|---|
| **Ship (Ramp)** | Items offloaded from the ship to the pier | Items | Constant |
| **JLTI** | Maintenance/operations check of the gear | Mechanics | Exponential |
| **POG** | Accountability of all gear coming off of the ship | Inspectors | Exponential |
| **Dis/Ass** | Disassociation/Association of gear on PEI's | Material Handlers | Exponential |
| **MCC** | Where vehicles are rearranged according to using unit | Coordinators | Exponential |
| **Convoy AAOE** | Convoys that transit to the AAOE's | Drivers | Exponential |
| **AAOE** | Using Units receiving and inspecting gear coming to them | Receivers | Exponential |

Table 3. The above table shows the Scenario 1 List of Event Graphs with Descriptions, Constraints, and Time Distributions.

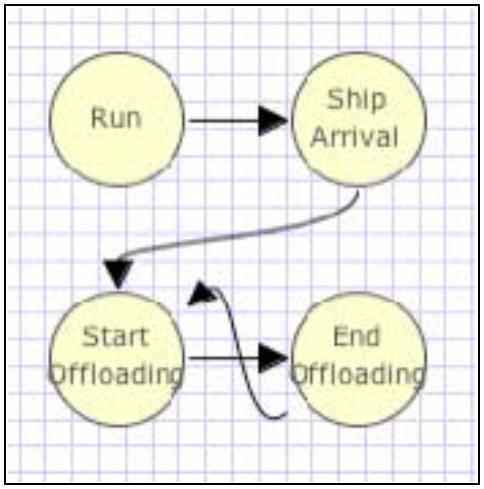



Figure 29. This is the ship offload event graph for the base scenario, Scenario 1.

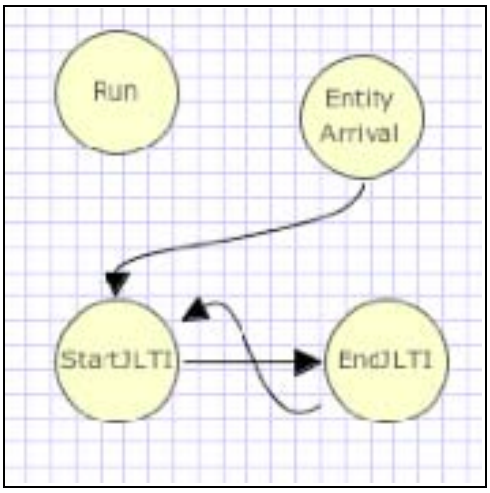



Figure 30. This is the event graph for the JLTI for Scenario 1.

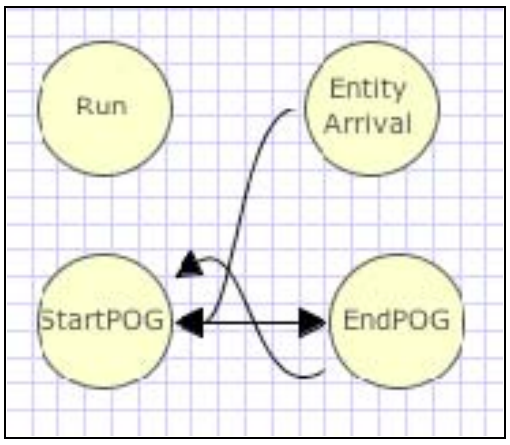



Figure 31. This is the Scenario 1 Port Operations Group accountability event graph.

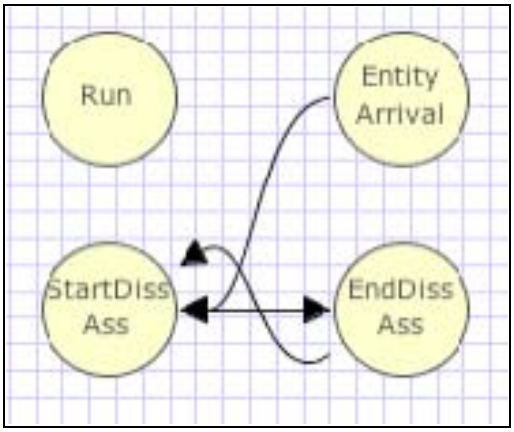



Figure 32. This is the Scenario 1 Assembly/Disassembly Lot event graph.

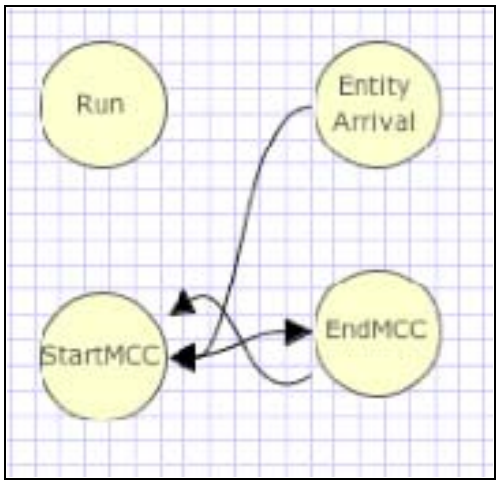



Figure 33. This is the Scenario 1 Movement Control Center event graph.

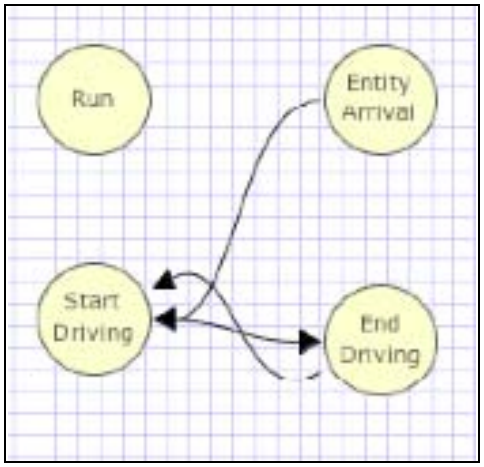



Figure 34. This is the Scenario 1 Convoy to the Arrival and Assembly Operations Element event graph.

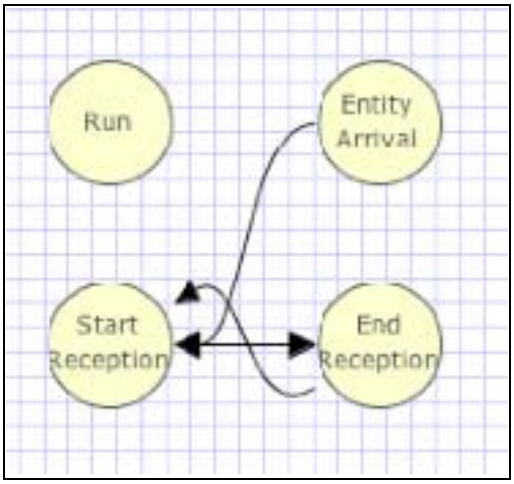



Figure 35. This is the Scenario 1 Arrival and Assembly Operations Element Reception event graph.

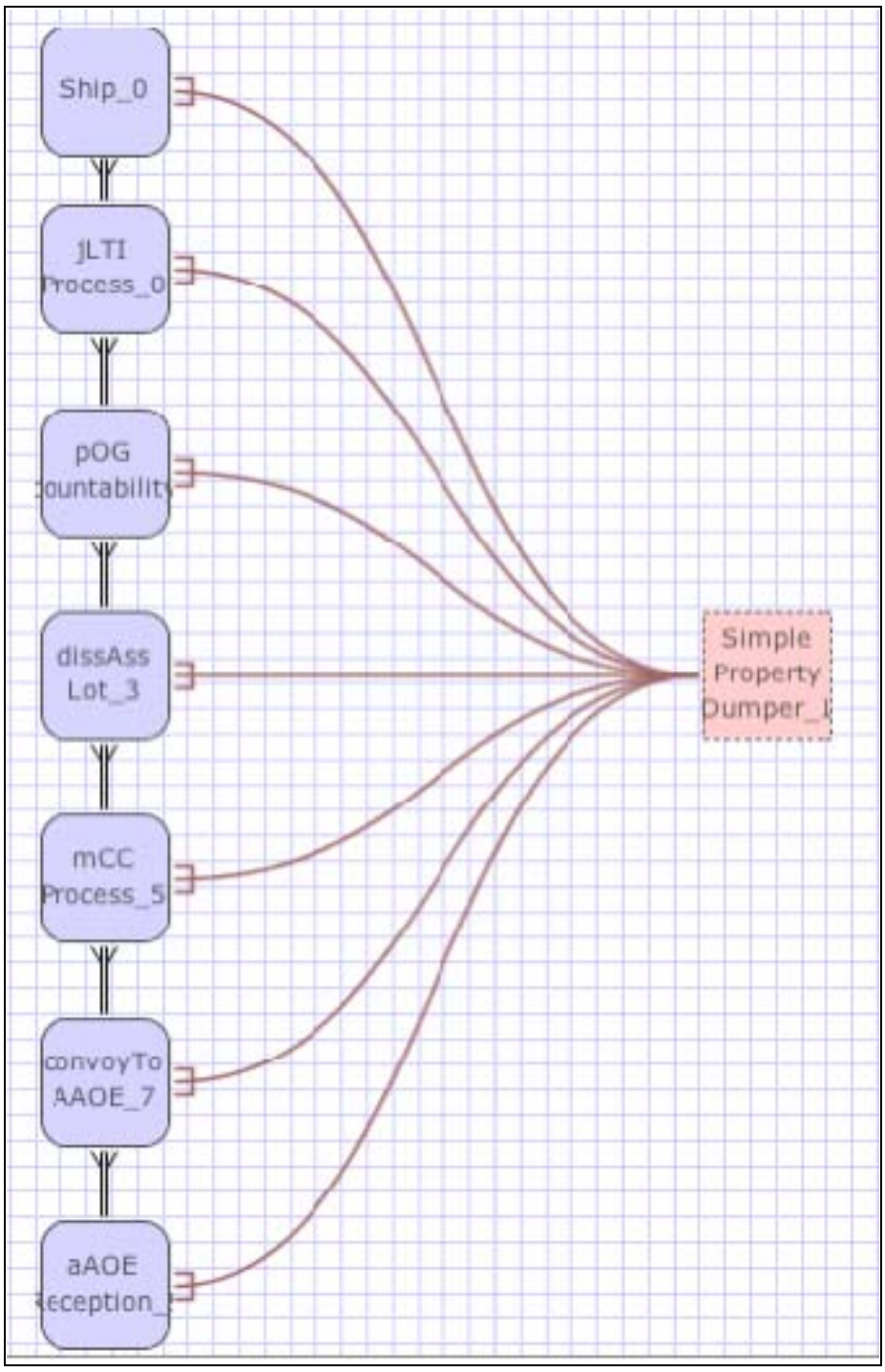



Figure 36. This is the Assembly Editor for Scenario 1.

### *b. Scenario 2*

The second scenario builds more complexity onto the base scenario. Several features of this scenario must be considered. The second scenario adds more complex distributions for all time-dependent tasks, as well as a transition event graph assembly that allows the scenario as a whole to account for all of the movement of the Principle End Items throughout the entire process. The changes in time distribution choices, as well as the addition of the transition event, are shown in Table 4.

| ***Process*** | ***Description*** | ***Constraint*** | ***Time Distribution*** |
|---|---|---|---|
| **Ship (Ramp)** | Items offloaded from the ship to the pier | Items | Triangular |
| **JLTI** | Maintenance/operations check of the gear | Mechanics | Triangular |
| **POG** | Accountability of all gear coming off of the ship | Inspectors | Triangular |
| **Dis/Ass** | Disassociation/Association of gear on PEI's | Material Handlers | Triangular |
| **MCC** | Where vehicles are rearranged according to using unit | Coordinators | Triangular |
| **Convoy AAOE** | Convoys that transit to the AAOE's | Drivers | Triangular |
| **AAOE** | Using Units receiving and inspecting gear coming to them | Receivers | Triangular |
| **Transition** | Drivers transfer items from one place to another | Receivers | Triangular |

Table 4. The above table shows the Scenario 2 List of Event Graphs with Descriptions, Constraints, and Time Distributions.

This black box approach enables planners to input the number of ships to be offloaded, the numbers and types of equipment to be offloaded, and the numbers of personnel at each location throughout the process, and determine how long a given scenario will take. Moreover, planners can modify parameters to fine-tune throughput management, allowing for better economy of force and proving this simulation's flexibility.

### *c. Summary*

The final scenario signals the end state in simulating the process. The image below depicts the idea of such a ground truth. The final scenario uses appropriate probability distributions for each time-dependent task based on relevant data collected from previous offloads, as well as distances and locations of the respective offload. The final scenario also takes into account each type of vehicle based on its TAMN, as well as the different amounts of time it takes each type of vehicle to complete each process. Again, as each scenario is built upon its predecessor, it becomes more complex and closer to the truth of the process on the ground.

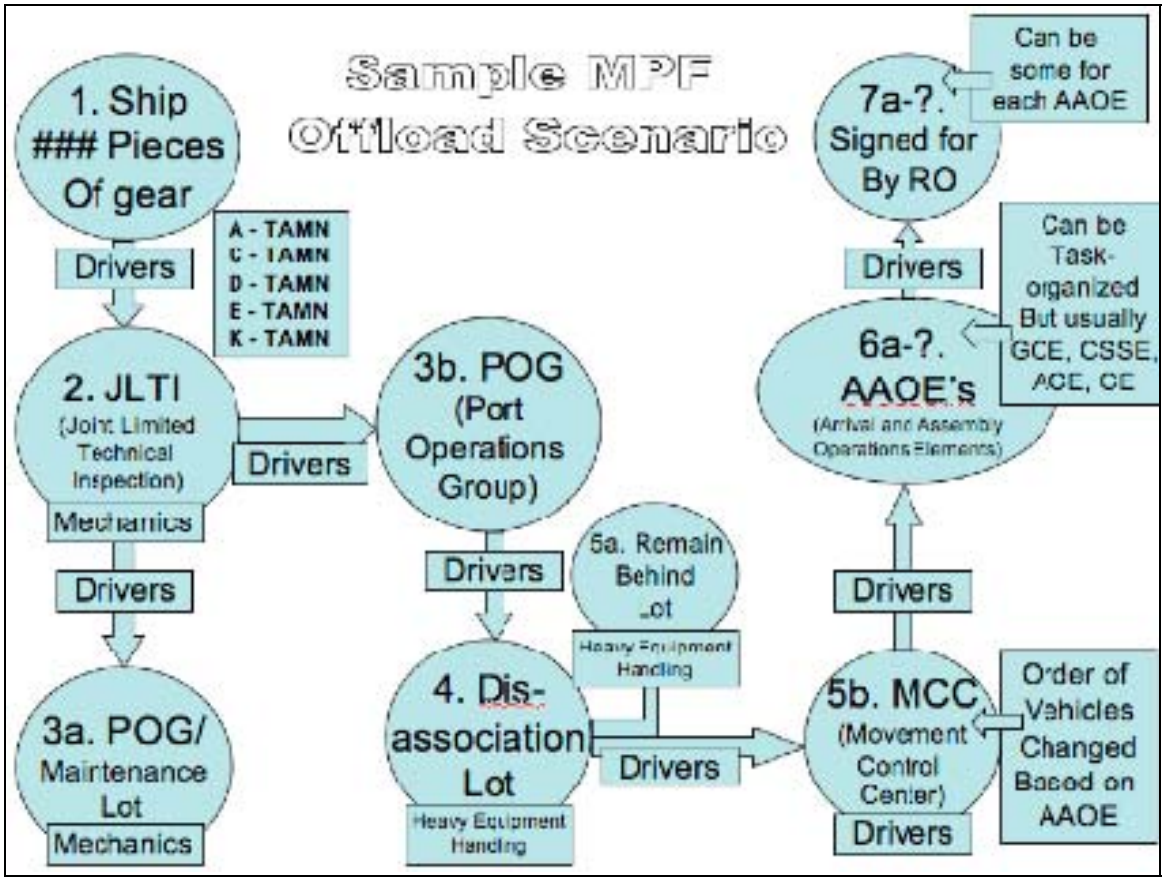



Figure 37. A replication of Figure 8, this depiction of the MPF Offload flow illustrates the final scenario, the end state in Extreme Programming.

THIS PAGE INTENTIONALLY LEFT BLANK

# V. SIMULATION RESULTS AND ANALYSIS

## A. INTRODUCTION

The results achieved by the simulations demonstrate the viability of the DES approach to this logistical process. The Design of Experiments (DOE) panel provides a combined look at all parameters of interest. Trace statements produced as execution output enable close checking that all interactions proceed sequentially and satisfactorily.

## B. DESIGN OF EXPERIMENTS (DOE)

The following image portrays the Design of Experiments, or DOE, for the simulation, and is generated by Viskit. This viewer proves extremely helpful in the overall analysis of parameters entered into the system, their type, and their values, not to mention time distributions associated with each.

| SimEntity/Parameter name | Type | Value |
|---|---|---|
| **Ship** | **Ship** | |
| numberOfItems | int | 10 |
| Ship_2 | simkit.random.RandomVariate | |
| Ship_2_1 | String | Triangle |
| Ship_2_2 | Object[] | |
| Ship_2_2_1 | Object | 3 |
| Ship_2_2_2 | Object | 6 |
| Ship_2_2_3 | Object | 4 |
| **jLTIProcess** | **JLTIProcess** | |
| totalNumberOfMechanics | int | 5 |
| jLTIProcess_2 | simkit.random.RandomVariate | |
| jLTIProcess_2_1 | String | Triangle |
| jLTIProcess_2_2 | Object[] | |
| jLTIProcess_2_2_1 | Object | 13 |
| jLTIProcess_2_2_2 | Object | 20 |
| jLTIProcess_2_2_3 | Object | 15 |
| **pOGAccountability** | **POGAccountability** | |
| totalNumberOfInspectors | int | 3 |
| pOGAccountability_2 | simkit.random.RandomVariate | |
| pOGAccountability_2_1 | String | Triangle |
| pOGAccountability_2_2 | Object[] | |
| pOGAccountability_2_2_1 | Object | 1 |
| pOGAccountability_2_2_2 | Object | 4 |
| pOGAccountability_2_2_3 | Object | 2 |
| **disAssocLot** | **DissAssLot** | |
| totalNumberOfMaterialHandlers | int | 5 |
| disAssocLot_2 | simkit.random.RandomVariate | |
| disAssocLot_2_1 | String | Triangle |
| disAssocLot_2_2 | Object[] | |
| disAssocLot_2_2_1 | Object | 8 |
| disAssocLot_2_2_2 | Object | 15 |
| disAssocLot_2_2_3 | Object | 10 |
| **mCCProcess** | **MCCProcess** | |
| totalNumberOfCoordinators | int | 10 |

Table 5. This is a DOE of the MPF Offload Base Scenario, showing the time distributions of each time-dependent task and associated times to completion.

## C. STATISTICAL RESULTS OF THE DISCRETE EVENT SIMULATION

The statistical results in a verbose output show the simulation as an event list and how each event fires off chronologically. The stop time allows the user to view the time at each location in the process. As can be seen in the following replication statistics that are created, the stopping time for the simulation is 100.0 units (units refers to any particular time unit chosen, consistency prevailing). The first replication begins with a random seed from a random seed generator that ensures that the Random Variate is not

predictive. The event graphs populate a master event list that determines in which order universally things are to occur. This simulation results in having taken a set amount of time, and measures the average wait times for each event.

```
Replication Statistic(s) created
--------------------------------

Stopping at time: 100.0
Starting Replication #1 with random seed -914616643 for:
** Event List 0 -- Starting Simulation **
0.000   Run      <mpfoffload.Offload.1>
0.000   Run      <Ship_0>
0.000   Run      <jLTIProcess_0>
0.000   Run      <pOGAccountability_0>
0.000   Run      <dissAssLot_3>
0.000   Run      <mCCProcess_5>
0.000   Run      <convoyToAAOE_7>
0.000   Run      <aAOEReception_9>
100.000          Stop     <Simkit.Stop.15>
 ** End of Event List -- Starting Simulation **

Time: 0.0000  CurrentEvent: Run [1]
** Event List 0 --  **
0.000   Run      <Ship_0>
0.000   Run      <jLTIProcess_0>
0.000   Run      <pOGAccountability_0>
0.000   Run      <dissAssLot_3>
0.000   Run      <mCCProcess_5>
0.000   Run      <convoyToAAOE_7>
0.000   Run      <aAOEReception_9>
100.000          Stop     <Simkit.Stop.15>
 ** End of Event List --  **

numberOfRemainingItems: null => 10
Time: 0.0000  CurrentEvent: Run [2]
** Event List 0 --  **
0.000   Run      <jLTIProcess_0>
0.000   Run      <pOGAccountability_0>
0.000   Run      <dissAssLot_3>
0.000   Run      <mCCProcess_5>
0.000   Run      <convoyToAAOE_7>
0.000   Run      <aAOEReception_9>
0.000   ShipArrival      <Ship_0>
100.000          Stop     <Simkit.Stop.15>
 ** End of Event List --  **
```

numberOfAvailableMechanics: null => 2
queue: null => []
Time: 0.0000  CurrentEvent: Run [3]
** Event List 0 --  **
0.000  Run  <pOGAccountability_0>
0.000  Run  <dissAssLot_3>
0.000  Run  <mCCProcess_5>
0.000  Run  <convoyToAAOE_7>
0.000  Run  <aAOEReception_9>
0.000  ShipArrival  <Ship_0>
100.000  Stop  <Simkit.Stop.15>
 ** End of Event List --  **

numberOfAvailableInspectors: null => 5
queue: null => []
Time: 0.0000  CurrentEvent: Run [4]
** Event List 0 --  **
0.000  Run  <dissAssLot_3>
0.000  Run  <mCCProcess_5>
0.000  Run  <convoyToAAOE_7>
0.000  Run  <aAOEReception_9>
0.000  ShipArrival  <Ship_0>
100.000  Stop  <Simkit.Stop.15>
 ** End of Event List --  **

numberOfAvailableMaterialHandlers: null => 3
queue: null => []
Time: 0.0000  CurrentEvent: Run [5]
** Event List 0 --  **
0.000  Run  <mCCProcess_5>
0.000  Run  <convoyToAAOE_7>
0.000  Run  <aAOEReception_9>
0.000  ShipArrival  <Ship_0>
100.000  Stop  <Simkit.Stop.15>
 ** End of Event List --  **

numberOfAvailableCoordinators: null => 6
queue: null => []
Time: 0.0000  CurrentEvent: Run [6]
** Event List 0 --  **
0.000  Run  <convoyToAAOE_7>
0.000  Run  <aAOEReception_9>
0.000  ShipArrival  <Ship_0>
100.000  Stop  <Simkit.Stop.15>
 ** End of Event List --  **

```
numberOfAvailableDrivers: null => 5
Time: 0.0000  CurrentEvent: Run [7]
** Event List 0 --  **
0.000   Run     <aAOEReception_9>
0.000   ShipArrival     <Ship_0>
100.000         Stop    <Simkit.Stop.15>
 ** End of Event List --  **

numberOfAvailableReceivers: null => 4
queue: null => []
Time: 0.0000  CurrentEvent: Run [8]
** Event List 0 --  **
0.000   ShipArrival     <Ship_0>
100.000         Stop    <Simkit.Stop.15>
 ** End of Event List --  **

Time: 0.0000  CurrentEvent: ShipArrival [1]
** Event List 0 --  **
0.000   StartOffloading         <Ship_0>
100.000         Stop    <Simkit.Stop.15>
 ** End of Event List --  **

numberOfRemainingItems: 10 => 9
Time: 0.0000  CurrentEvent: StartOffloading [1]
** Event List 0 --  **
1.100   EndOffloading           {item.1 [0.0000,0.0000]}        <Ship_0>
100.000         Stop    <Simkit.Stop.15>
 ** End of Event List --  **

queue: [] => [item.1 [0.0000,0.0000]]
Time: 1.1000  CurrentEvent: EndOffloading           {item.1 [0.0000,0.0000]} [1]
** Event List 0 --  **
1.100   StartOffloading         <Ship_0>
1.100   StartJLTI       <jLTIProcess_0>
100.000         Stop    <Simkit.Stop.15>
 ** End of Event List --  **

numberOfRemainingItems: 9 => 8
Time: 1.1000  CurrentEvent: StartOffloading [2]
** Event List 0 --  **
1.100   StartJLTI       <jLTIProcess_0>
2.200   EndOffloading           {item.2 [1.1000,1.1000]}        <Ship_0>
100.000         Stop    <Simkit.Stop.15>
 ** End of Event List --  **
```

numberOfAvailableMechanics: 2 => 1
queue: [item.1 [0.0000,0.0000]] => []
Time: 1.1000  CurrentEvent: StartJLTI [1]
** Event List 0 --  **
2.200  EndOffloading  {item.2 [1.1000,1.1000]}  <Ship_0>
6.456  EndJLTI  {item.1 [0.0000,0.0000]}  <jLTIProcess_0>
100.000  Stop  <Simkit.Stop.15>
 ** End of Event List --  **

queue: [] => [item.2 [1.1000,1.1000]]
Time: 2.2000  CurrentEvent: EndOffloading  {item.2 [1.1000,1.1000]}

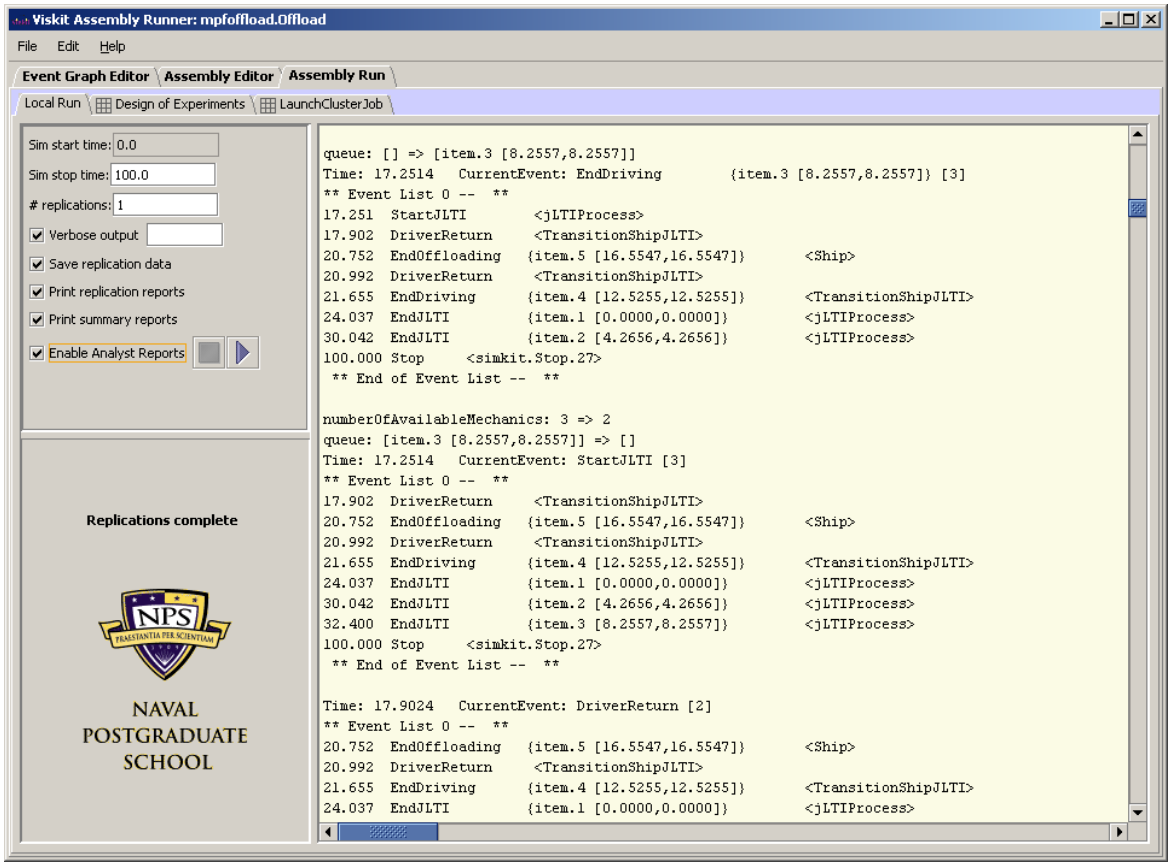



Figure 38.  The above captures the statistical results in Viskit for the Base Scenario.

## D. TIME DISTRIBUTIONS

The following probability distribution functions for time-duration variability must be considered in an effort to create a more realistic and valid simulation. The following list addresses time-dependent tasks that fall under the realm of this offload process that must be determined for the simulation:

-Time to LO/LO containers

-Time to LO/LO HMMWV's

-Time to RO/RO wheeled vehicles

-Time to RO/RO tracked vehicles

-Time to offload EAF (Expeditionary Air Field)

-Time to offload Fleet Hospitals

-Time to perform JLTI's

-Time to perform disassembly/assembly operations

-Time to coordinate vehicles inside the MCC according to AAOE

-Times to drive convoys to each AAOE

This list is by no means all-inclusive, but merely serves to give an idea of the time considerations that must be made as inputs to the simulation. A variety of probability distribution functions (PDF's) are now examined in order to consider which are best suited to represent the statistical variations occurring in the real world.

### 1. Exponential Distribution

Describing the time between events in a Poisson process, which the exponential distribution is "memory-less" distribution—that is, if one were to approach a mechanic 15 minutes after he begins his work, and this particular job is supposed to take "about an hour," and ask him, "How much longer will this job of yours take?" Instead of saying, "another 45 minutes," he would say, "about an hour." The reason for this is that according to the distribution, he has no recollection as to how long he has been working on this task, even if he had been working on it for 59 minutes. This distribution, although

simple, does not capture well the nature of a distribution for task completion. It is used, however, in the first base scenario because it only uses one parameter as its input and is a simpler distribution.

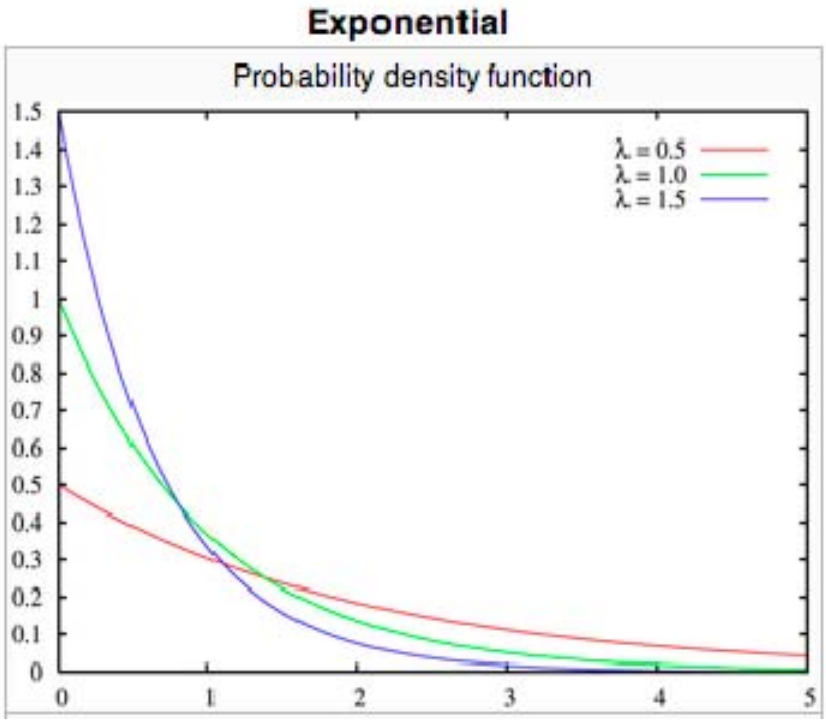



Figure 39. This is a depiction of the Exponential probability distribution function.

### 2. Gamma Distribution

The Gamma Distribution, although seemingly ideal for task accomplishments, is complex and hard to define. The shape of the distribution is ideal for task accomplishment. With a steep forward leaning curve, there is more of a probability that a given task will take the mean time given or a little faster than the mean time given. There is a significantly lower probability that a given task will take much longer than the mean time given, though there is nevertheless a probability of that. For example, if it normally takes someone thirty minutes to drive to work, it might occasionally take them twenty or twenty-five minutes. Once in a great while, due to an accident or some unforeseen occurrence, it may take that someone as long as three hours to get there, hence the gamma distribution. Though ideal, this distribution is much harder to measure and thus more unrealistic.

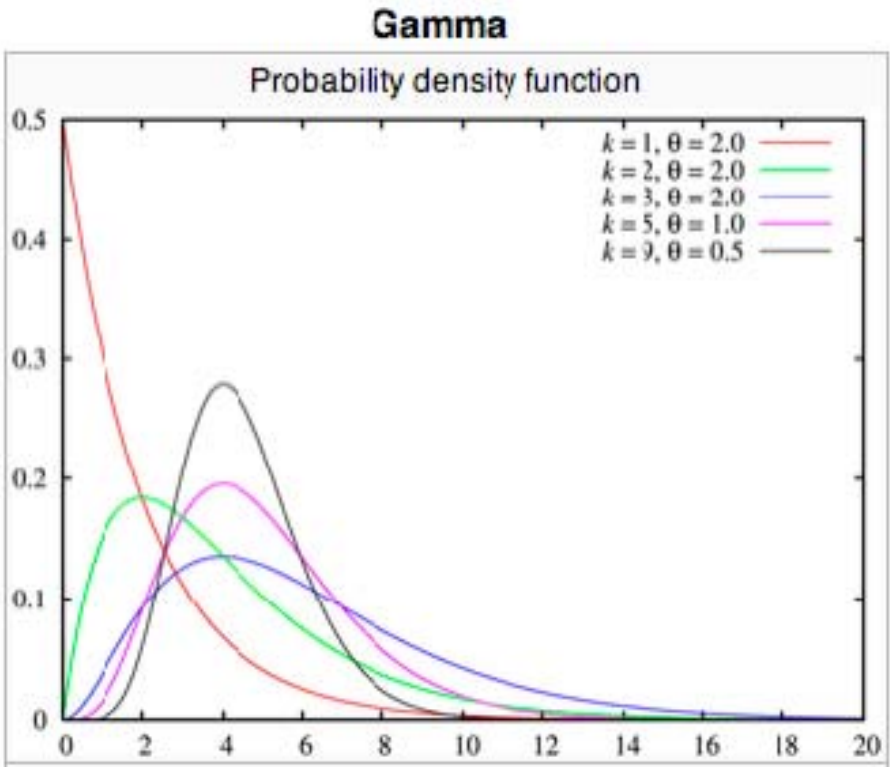



Figure 40. This is a depiction of the Gamma probability distribution function.

### 3. Triangular Distribution

The triangular distribution is the optimal mixture of a realistic distribution for time-dependent tasks and a relatively feasible way to determine the appropriate parameters to input. Rather than having to determine the parameters that would give a distribution curve its appropriate shape, the inputs for a triangular distribution are the mean time expected for a given task to be accomplished, the minimum time, and maximum time. A mixture of data collection and subject matter expert input can approximate these three parameters.

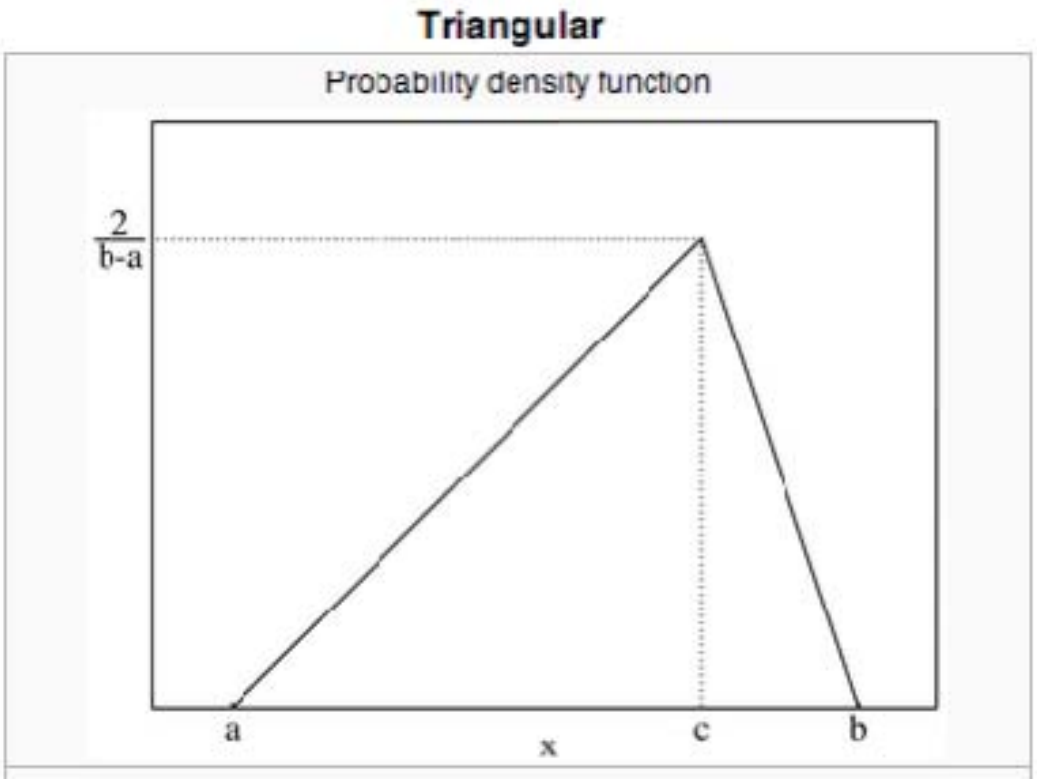



Figure 41. This is a depiction of the Triangular probability distribution function.

## E. FUTURE REPORT DEVELOPMENT

Future results interface with a statistical package and analyze parts of the process that take too long, that might hinder the throughput of gear and equipment. Needed improvements to statistical reports include the following:

-Graphical results that compare average wait times at individual event nodes

-Easier to read, more organized output

-Selectable level of detail describing Design of Experiments (DOE)

-Executive summary and briefing slide set

## F. SUMMARY

In conclusion, the second scenario accounts for all of the movement of the equipment as it enters the process until it completes the process. This complete simulation is analyzed to determine the average wait time at each location, whose parameters can be modified to aid in more maximized throughput through the system.

THIS PAGE INTENTIONALLY LEFT BLANK

# VI. CONCLUSIONS AND RECOMMENDATIONS

## A. CONCLUSIONS

Viskit proves itself a viable option in converting the logistical process that is the MPF offload into a simulation, whose parameters can be more scientifically determined. Instead of following an arbitrary set of recommended numbers of personnel at each location in an offload, this simulation can be used to determine how many personnel will maximize the throughput of the system.

## B. RECOMMENDATIONS FOR FUTURE WORK

Logistics operations such as convoy operations, tracking of people and equipment from one mode of transportation to another can be applied to the same technologies for tracking and visibility.

### 1. Future Work Intended for Current Simulation

After the simulation is validated, a visualization tool such as X3D enables visual tracking of real-time or simulated movement of the following as they traverse through the MPF Offload Process:

a. Rolling Stock: Tanks, HMMWV's, AAV's, MTVR's, LVS's, Dozers, Cranes, RTCH's, etc.

b. Cargo to a specified level of detail: Trailers, Water Bulls, Fuel and Water Containers, Containers, etc.

c. People: Drivers, Assistant Drivers, Contracted Vehicles with drivers, Mechanics, etc.

The goal in the visualization is to make everything simple, color-coded by using unit, so that respective commanders and operations officers can see where their equipment is at any given time. Interoperability with RFID and BFT can further aid in making the model more than a simulation of predefined parameters; these interoperability tools enable real-time tracking.

### *a. Methodology*

Convert data into XML, create and use SAVAGE models of vehicles, cargo, personnel, port areas, create time-stepped data updates to track the flow of all people and things through the network. Convert this simulation into an analytical tool in order to use optimization, autonomous agents design, and Viskit scenario Discrete Event Simulation in an attempt at improving the efficiency of the system. Minimize the number of drivers and mechanics needed to operate the system. Recommend optimal security emplacement to secure the network.

### *b. End State*

The end state for this simulation is to be able to input the data into the system, such as how the ships are loaded (ICODES, from Blount Island Command) and the particular port into which this or these ship(s) is/are loading and have "it" spit out a scenario based on these parameters, giving the user information as to how many drivers and other personnel needed for the offload, how long it will take to accomplish, etc.

## 2. Beach Operations Groups

In the case of the BOG, a Model can be built using X3D and X3D Earth, with models from the SAVAGE Archive. Once the data captured from the BOG can be attached to the models, a simulation can be built using either Viskit or Simkit and the entire operation, from the landing support Marines hitting the beach and setting up beach panel markers, essentially "marking the beach" so that the ships off the coast know where to bring what, to the creation of a workspace on the beach by earthmoving equipment, to the movement and further deployment of supplies and possibly even personnel. The model (simulation) may finish as routine re-supply convoys support the forward forces, or as a camp is established, or even so far as the regeneration and redeployment of all gear and personnel back to sea is accomplished. Such a model for a BOG can be used in training at EWTGLANT in the teaching of amphibious raids, but also in planning and perhaps some analysis, which actually depends on how validate-able the model might become.

### 3. Logistics Operations Centers

Logistics Operations Centers (LOC's) contain a whirlwind of active information that changes constantly. To "XML-ize" such an operations center, add visualization, add artificial intelligence, and add anything else that might aid in the automation of such a place, can only serve to bring a slice of order to this pie of chaos. All aspects of modeling and simulation are direly needed, ironically, to add simplicity to this complex beast of fulfilling logistical requirements, tracking everyone else's, and taking care of their own.

### 4. X3D Visualizations

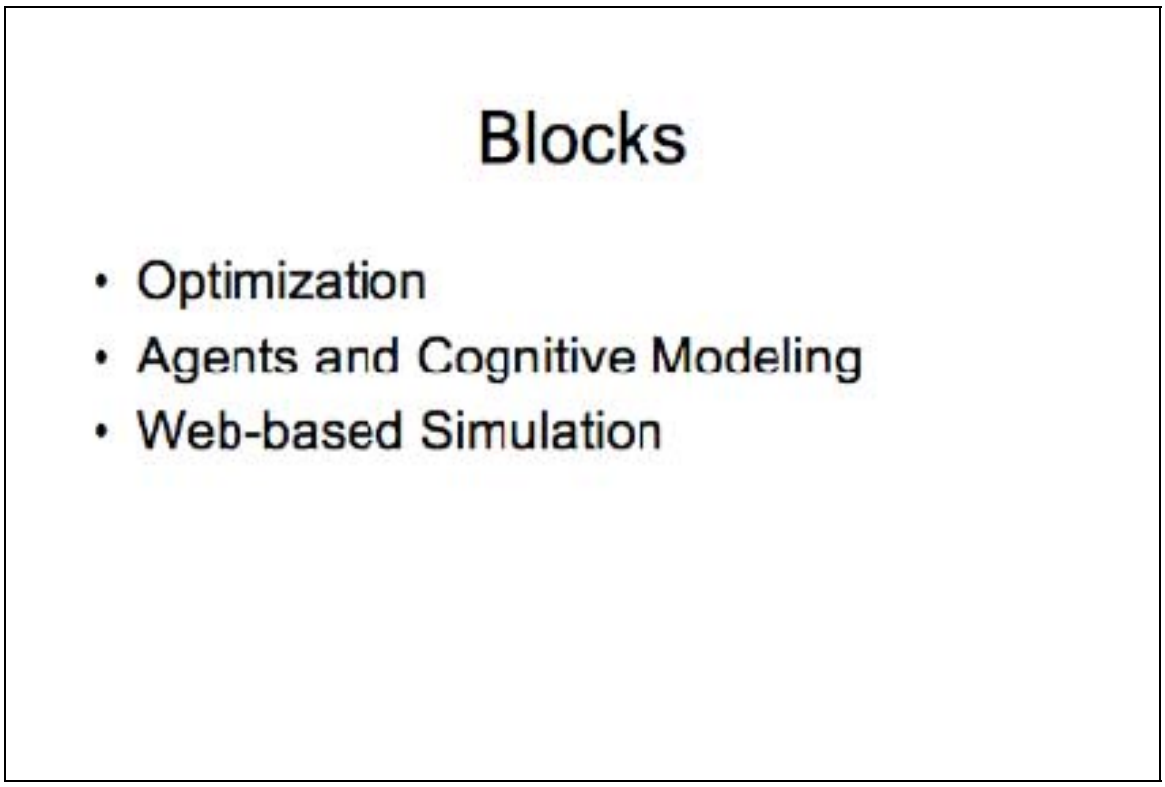
The incorporation of X3D visualization into the overall design of developing a Common Operating Picture (COP) for the Arrival and Assembly Operations Group is essential to the completion of a comprehensive operations center. The incorporation of the MPF Offload process into the following X3D tools proves instrumental in the development of a visual capability that has up to this point been lacking:

- Savage Studio integration
- X3D Models in Savage with SMAL metadata
- Viskit behaviors in version control
- Catalog creation for all behaviors in a manner similar to Savage catalog creation.

THIS PAGE INTENTIONALLY LEFT BLANK

# VII. APPENDICES

## A. EXECUTIVE BRIEF



# Beach Operations

- Expeditionary Warfare Demonstrator (EWD)
  - Shortfall identified, needs updating, built in 1948!
  - Commander Saat, EWTGLANT
    - Particularly interested in models that can show the ship-to-shore transition well
  - Thesis Opportunity
- 3D Model of an Amphibious Raid
  - What you can do virtually that you cannot do physically-
    - Can show ranges of sensors, weapons systems, etc.
    - Flexibility of scenarios

# Beach Operations

- Expeditionary Warfare Demonstrator (EWD)
  - Shortfall identified, needs updating, built in 1948!
  - Commander Saat, EWTGLANT
    - Particularly interested in models that can show the ship-to-shore transition well
  - Thesis Opportunity
- 3D Model of an Amphibious Raid
  - What you can do virtually that you cannot do physically-
    - Can show ranges of sensors, weapons systems, etc.
    - Flexibility of scenarios

## More on Port Operations

- Ship - POG - MCC - Road Networks - Using Units
- End state - An operations officer or Using Unit representative (and sometimes the CO) can walk into the AAOG (arrival and assembly operations group) Command Center and see a 3D representation of where everything is, potentially color-coded by unit
- Used for tracking, optimizing, analyzing, and potentially for training

## Technical Considerations

- Interoperability with RFID (Radio Frequency Identification) - bar-coded equipment
- Interoperability with BFT (Blue Force Tracker) - Convoy Tracker, goes with each significant convoy

# Convoy Operations

- Same Concept as with Port Operations - Convert the data into XML - into VISKIT - into a 3D track-able representation of the process
- Making some order of chaos
  - Handling Support requests
  - Where can AI help?
    - Converting Support requests into Planned Convoys
    - Current Operations - Prepping Convoys for the Convoy Commander/Assistant Convoy Commander - Training
    - Analyzing and tracking Road Networks (Some Counter IED Implications)

# Process

- 1- VISKIT Discrete Event Simulation of Scenario
- 2- Complex Adaptive System Design of Scenario
- 3- Optimization Formulation of Scenario
- 4- Convert Data into XML
- 5- Capture Relevant Terrain Data in X3D - Port, Roads, Camps
- 6- Build/Use Relevant Models in X3D
- 7- Put it all together… Visualization
- 8- Capture Relevant Statistics and Compare

## B. THREE POLICY-LEVEL ACTIONS TO HELP PROMOTE OPTIMUM DOD USE OF OPEN SOURCE SOFTWARE

### 1. Create a "Generally Recognized As Safe" Oss List

This list would provide quick official recognition of OSS applications that are (a) commercially supported, (b) widely used, and (c) have proven track records of security and reliability

### 2. Develop Generic, Infrastructure, Development, Security, & Research Policies

The DoD should develop generic policies both to promote broader and more effective use of OSS, and to encourage the use of commercial products that work well with OSS.

### 3. Encourage Use of FOSS to Promote Product Diversity

Acquisition diversity reduces the cost and security risks of being fully dependent on a single software product, while architectural diversity lowers the risk of catastrophic cyber attacks based on automated exploitation of specific features or flaws of very widely deployed products.

Acceptable OSS Licenses

| Name | Originating Organization |
|---|---|
| Mozilla Public License (MPL) | Netscape Communications |
| General Public License (52%) | Free Software Foundation |
| BSD License (6%) | University of California |
| Artistic License (1%) | Larry Wall, creator of Perl |
| MIT/X Consortium License | MIT/X Consortium |

Table 6. This is a table of Open Source Solutions.

## C. NETWORK INTERDICTION PROJECT – MINIMIZING THE RISK FOR AN MPF OFFLOAD

Optimization is a technique that also attempts to maximize throughput of a system. In this UNCLASSIFIED Network Flow problem, the road network of an MPF offload can be used in a network flow problem to determine where to best emplace security personnel, to maximize the security of the same process. Oftentimes, the AAOE's are far away from the port where the offload takes place. The map below depicts such a road network, branching out from a Movement Control Center to three respective AAOE sites.

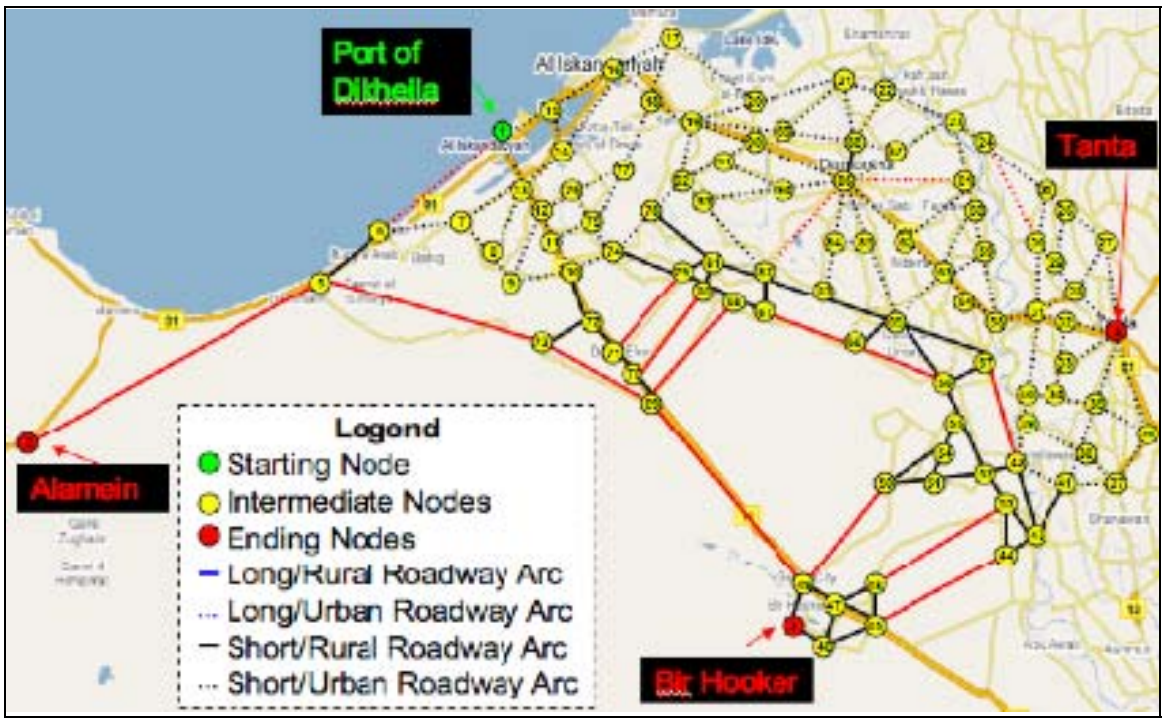



Figure 42. This is a Network Flow Diagram of the northern Egypt road network.

This network represents a test case for the movement and deliveries of equipment and rolling stock from a port where it will have been off-loaded via MPF (Maritime Prepositioning Force) Shipping to respective using units where it will be used in follow on missions. This operation in itself is not mission dependent; as operational logistics, it enables expeditious accomplishment of any given mission. The problem deals with

providing optimal security—the optimal use of limited security forces to protect the lines of supply, the supply routes, ensuring better economy of force, to ensure that the mission is best supported.

The global prepositioning of ships proves itself to be one of the quickest and most efficient ways to bring equipment and rolling stock abroad to foreign ports for use by the United States military for a plethora of missions, from peacekeeping to disaster relief to winning the nation's wars. While current techniques for providing security to ensure that supplies and rolling stock are delivered in a timely manner, these techniques do not take into account the optimization of the emplacement of friendly security forces.

Moreover, pilfering and stealing of such rolling stock and equipment is an issue. In an environment where the threat is asymmetric, a diligent risk analysis coupled with the optimization of minimum risk analysis can help to pinpoint otherwise unintuitive places to position security forces to best protect the delivery of these essential pieces of rolling stock. The idea of marrying up gear with people in country makes this evolution vital to mission accomplishment.

Currently security accompanies the convoys that move from the port to the respective using unit, and security forces tie themselves primarily to the port. Security detachments are not currently emplaced in areas that use operational analysis of the network. While security forces are emplaced at tactically beneficial points, these are not the optimal places to position security forces to counter the possibility of attacks by the given enemy based on given risk analysis.

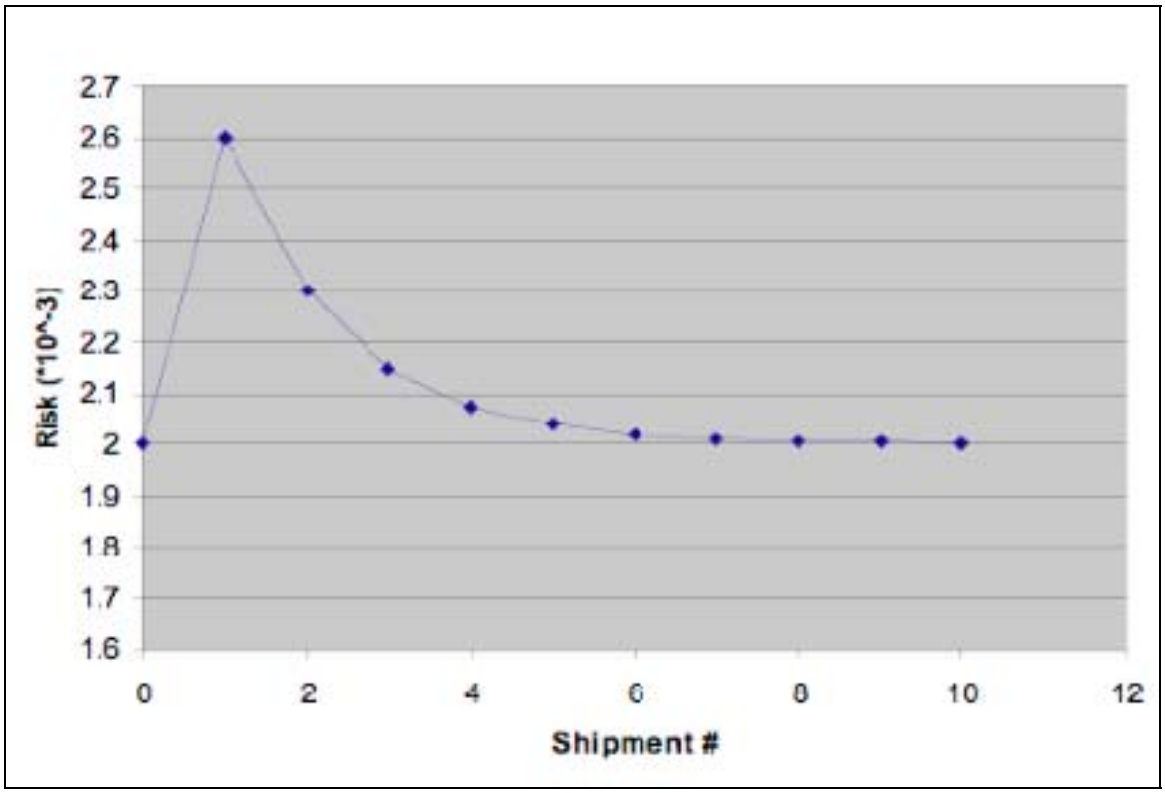


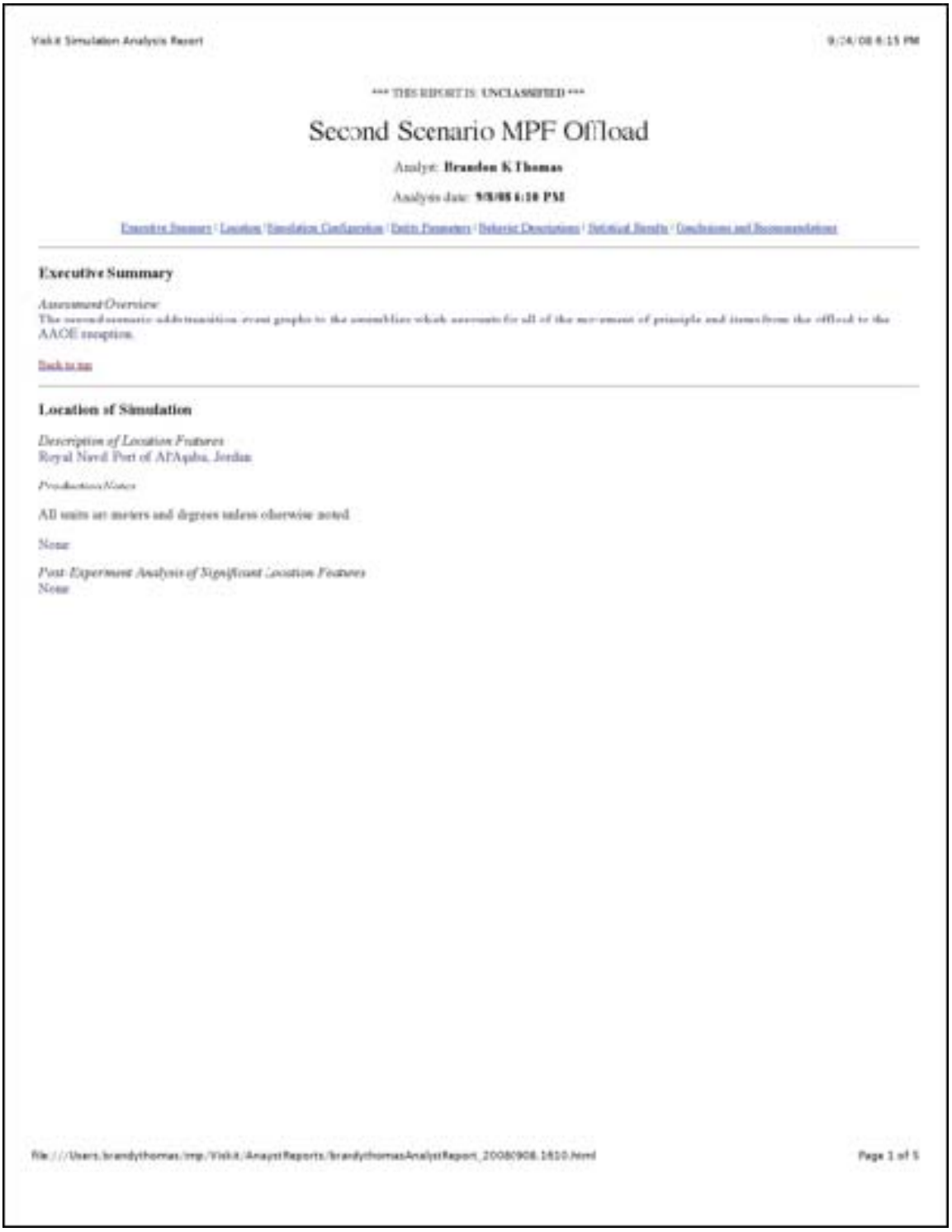

Figure 43. This shows a risk analysis based on the number of convoys/shipments.

We created a multiple-destination, minimum risk analysis based on a somewhat arbitrary analysis of the risk in a test-case network, in this case, an MPF offload in Egypt. This test case exemplifies Exercise Bright Star, a nation-building coalition exercise that provides a good snapshot of user-friendly and analysis-friendly offloads. Although current Operation Iraqi Freedom missions have superceded the existence of Exercise Bright Star for now, nothing precludes such an exercise resuming in the future, and the geography of the region provides an excellent opportunity for the analysis of such a network. We assigned risks of attack based on whether the road arcs were in urban or rural environments, whether they were long or short road arcs. We also assigned a set probability of success by the enemy given an attack along an arc. We converted those probabilities into additive risk indices, and were able to use GAMS to determine optimal attacks by the enemy, what would maximize our own risk, based on one attack, two attacks, three attacks, and so forth and so on by the enemy. The deliverable for us would

be optimal locations to emplace security forces depending on how many forces are available. These locations were not prioritize-able.

The resultant GAMS output showed the optimal arcs to interdict (thereby secure) given one attack by the enemy, and two attacks, and so on, independently. Through simple observance of the maps based on the given numbers of attacks, the risks increase in a logical fashion in accordance with the given arcs that are interdicted.

## D. VISKIT ANALYST REPORT



Viskit Simulation Analysis Report 9/14/08 6:15 PM

*** THIS REPORT IS: UNCLASSIFIED ***

# Second Scenario MPF Offload

Analyst: **Brandon K Thomas**

Analysis date: **9/8/08 6:10 PM**

Executive Summary | Location | Simulation Configuration | Entity Parameters | Behavior Descriptions | Statistical Results | Conclusions and Recommendations

## Executive Summary

*Assessment Overview*
The second scenario edit/transition event graphs to the assemblies which accounts for all of the movement of principle and items from the offload to the AAOE reception.

Back to top

## Location of Simulation

*Description of Location Features*
Royal Naval Port of Al'Aqaba, Jordan

*Production Notes*

All units are meters and degrees unless otherwise noted

None

*Post-Experiment Analysis of Significant Location Features*
None

file:///Users/brandythomas/mp/Viskit/AnalystReports/brandythomasAnalystReport_20080908.1810.html Page 1 of 5

Figure 1: 2D Overview of Simulation Study Area

Back to top

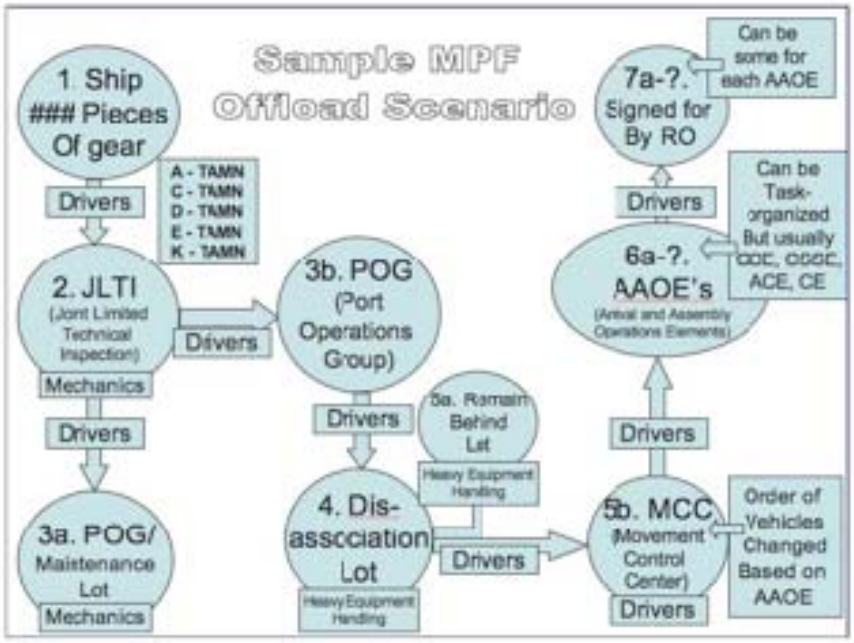



Figure 2: Digital Chart View of Study Area

Back to top

**Simulation Configuration: Viskit Assembly**

The simulation is defined by the Viskit Assembly which collects, lists, initializes and connects all participating entity models within a single scenario. The assembly is then ready for repeated simulation replications, either for visual validation of behavior or statistical analysis of Measures of Effectiveness (MoEs).

*Assembly Design Considerations*
Transition Event Graphs were added to connect each of the Process Event Graphs to account for all movement

*Production Notes*

All units are meters and degrees unless otherwise noted.

None

*Post-Experiment Analysis of Simulation Assembly Design*
None

**Summary of Simulation Entities**

| Simulation Entity | Behavior Definitions |
|---|---|

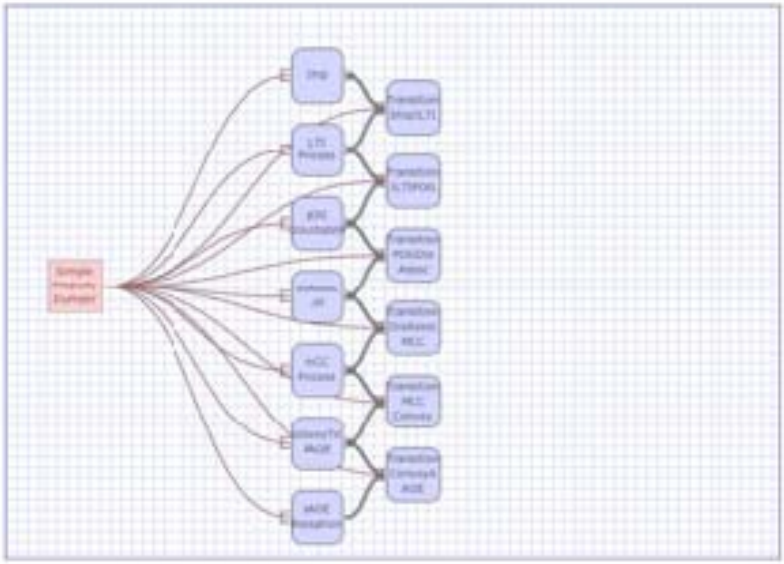


Figure 3: Simulation Assembly Containing all Simulation Entities for this Scenario Experiment

Back to top

### Entity Initialization Parameters for this Simulation Assembly

Initialization parameters are applied to individualize generic behavior models. These parameters customize the event-graph models.

*Entity Parameters Overview*
None

### Behavior Descriptions

*Description of Behavior Design*
None

*Post-Experiment Analysis of Entity Behaviors*
None

### Statistical Results

*Description of Expected Results*
None

*Analysis of Experimental Results*
None

Summary Statistics section: Primary Measures of Effectiveness (MoEs) / Measures of Performance (MoPs) and corresponding statistical plots

### Summary Report

| Entity | MoE / MoP | # Replications | Min | Max | Mean | StdDev | Variance |
|---|---|---|---|---|---|---|---|

Back to top

## Conclusions and Recommendations

*Conclusions*
The Second scenario succeeded in updating time distributions and accounting for the movement of all entities.

*Recommendations for Future Work*
The future scenarios must account for multiple types of items, or IEDs, and multiple ships and multiple AsOE reception locations.

Back to top

This report was autogenerated by the Vivkit Event Graph and Assembly modeling tool using Simkit discrete-event simulation (DES) libraries. Online at https://liana.nps.edu/Vivkit and https://liana.nps.edu/Simkit.

# LIST OF REFERENCES


Brutzman, D. (2008). Savage Defense Archive from https://savagedefense.nps.navy.mil/SavageDefense. Last accessed September 2008.

Brutzman, D. (2008). Viskit Home Page from http://diana.nps.edu/Viskit/. Last accessed September 2008.

Brutzman, D. (2008). X3D-Edit Home Page from https://savage.nps.edu/X3D-Edit Buss, A. (2008). Viskit Home Page from http://diana.nps.edu/Viskit. Last accessed September 2008.

Brutzman, D., & Daly, L. (2007). Extensible X3D Graphics for Web Authors. San Francisco: Morgan Kaufmann Publishers.

Buss, A. (2002). Component Based Simulation Modeling with Simkit. Proceedings of the 2002 Winter Simulation Conference, pp. 243-249.

Davidsson, P., Henesey, L., Ramstedt, L, Tornquist, J., & Wernstedt, F. Agent-Based Approaches to Transport Logistics.

MCO P3000.17A. Maritime Prepositioning Force Planning and Policy Manual (MPF Planning and Policy Manual).

MCWP 3-32/NTTP 3-02.3M. Maritime Prepositioning Force Operations (2004, January).

Ntuen, C.A. (2005, December) Logistics Planning for Coalition Command and Control.

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. BGen D.G. Reist, USMC
   Marine Corps Installations and Logistics
   Washington, DC

4. LtCol Blizzard, USMC
   Blount Island Command
   Orlando, FL

5. Col J.D.Turlip, USMC
   Logistics Modernization Command
   Albany, GA

6. MPF Officer
   I Marine Expeditionary Force
   Camp Pendleton, CA

7. Don Brutzman
   Naval Postgraduate School
   Monterey, CA

8. Ed Lesnowicz
   Naval Postgraduate School
   Monterey, CA